NASA TM-86404

NASA Technical Memorandum 86404

CARE III MODEL OVERVIEW AND USER'S GUIDE (first revision)

S. J. Bavuso

P. L. Petersen

April 1985

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23665

PREFACE

The NASA Langley Research Center has been developing reliability modeling technology for over a decade (ref. 1). The culmination of that research is embodied in the Computer-Aided Reliability Estimation computer program, third generation (CARE III). The CARE III model design was codeveloped by Dr. Jack J. Stiffler, formerly of the Raytheon Company, and Salvatore J. Bavuso of the NASA Langley Research Center. Dr. Stiffler designed and implemented the original CARE III mathematical model and Lynn A. Bryant of the Raytheon Company designed and implemented the computer code (ref. 2). I (S. J. Bavuso) implemented the original CARE III specification and participated in its long term design and implementation. This version of CARE III was designated as version 3 and was delivered to NASA as a proof-of-concept implementation.

The Boeing Computer Services Company of Seattle, Washington, validated the CARE III model and computer code. In the process, BCS made major modifications to the CARE III model and code bringing CARE III to production status code for the CDC CYBER 170 series computers under Fortran IV. The validated upgraded version is called version 4 and was available through NASA's software dissemination center, COSMIC, as release 1 beginning August 1984. Subsequently, version 4 was further upgraded to version 5 and is now implemented in ANSI standard Fortran 77. Version 5 compiles and executes on the VAX-11/700 series computers (VAX-11 Fortran compiler), the CYBER 170 series computers (CDC Fortran V compiler), and should be compatible with other computers that will compile ANSI standard Fortran 77.

Version 5 (release 2) has been enhanced significantly and includes the following modifications: The input program module CAREIN was completely rewritten to include extensive input data checking and more friendly error messages. The fault tree algorithm was improved to increase long mission time computation accuracy and was made more autonomous with the deletion of the KWT control parameter. The CARE3 program module was modified to include a variable time step integrator as a user option for improving the computational accuracy of $P(t|\underline{\ell})$ and $Q(t|\underline{\ell})$ functions.

This document is structured for the reliability engineering community and is specifically targeted toward the fault-tolerant ultrareliable system application. Although the initial CARE III application was directed at the digital aircraft flight control systems application, the generality of the CARE III model makes it easily applicable to a variety of highly reliable systems. The document is organized into two main parts: the main body and the appendix. The beginner will undoubtedly need to look at both parts. More experienced users will usually work with only the first part and particularly section 3 which takes the form of a quick-look reference manual. Sections 1 and 2 provide a brief overview of the model for those users who are attempting to quickly determine if CARE III is applicable to their application. Section 4 addresses CARE III output and its interpretation.

The appendixes provide backup information particularly for first time users. Appendix A covers machine dependency information and CARE III generated error and warning message. Appendixes B thru F provide additional information on some important CARE III subtleties, and appendix G contains example problems and some CARE III runs. Additional examples appear in reference 3 which contains eight example problems, two of which appear in Appendix G.2 in this document.

A substantial effort was made to minimize the use of jargon, peculiar to experts involved in fault-tolerant research, in order to make this document more descriptive and, hence, intelligible to newcomers. An example illustrating the need for clarity of terminology involves the many uses of the word "coverage" in the literature. In order to minimize confusion over the use of this term, CARE III uses this term to generically mean the "covering" of a fault. In the CARE III sense, coverage is an output of CARE III rather than an input to the math model. Therefore, CARE III differs significantly from many reliability modeling approaches which require coverage probabilities as inputs. The difficulty with using coverage as an input is that the coverage probability derivation is left to the ingenuity (more often, engineering guesswork) of the analyst. CARE III provides a fault-handling model that computes coverage based on system measurable (or estimative) parameters - still a nontrivial task.

The use of the term coverage is minimally applied in this document to identify mathematical expressions. Instead, more descriptive terminology replaces it. Unfortunately, the developers of CARE III chose to use a freer style of coverage terminology which may present some difficulty to readers of references 2, 4, and 6. These reports describe the CARE III mathematical model. The authors feel that the inconsistency should be only a minor irritation to those who find it necessary to read the mathematical description documents.

A final word about this document... The authors feel that it represents a good introduction to CARE III; however, we have every expectation of improving the guide and offering the user community the opportunity to aid in its improvement.

Many individuals listed on the next page have made important contributions to the development of CARE III. I wish to acknowledge the main contributors and personally thank all the contributors for their part in bringing CARE III to its present status.

J. J. Stiffler[1] L. A. Bryant	Raytheon Company - Model designer. Raytheon Company - Code designer of CARE III, version 3.
L. A. Bryant	Sequoia Systems, Inc. - Validation and enhancement of CARE III, version 5.
B. L. Dove H. M. Holt A. O. Lupton	NASA Langley Research Center - Managerial support.
D. M. Rose R. E. Altschul J. W. Manke D. L. Nelson	Boeing Computer Services Company - Validation and enhancement of CARE III, version 4. D. M. Rose was coauthor to NASA TM 85810, CARE III Model Overview and User's Guide, June 1984.
K. S. Trivedi R. Geist	Duke University - Tutorial insight and initial evaluation.
P. L. Petersen	Kentron International - Validation, implementation, and enhancement.
Boeing Commercial Airplane Company, Seattle, Washington	Fault Tree Program, FTREE, used in CARE III version 4 input program.
Air Force Avionics Lab., Wright-Patterson AFB, Ohio	Cofunded CARE III model development with the Langley Research Center.

Salvatore J. Bavuso
CARE III Project Engineer

1. Now employed by Sequoia Systems, Inc.

TABLE OF CONTENTS

LIST OF FIGURES

1.0 INTRODUCTION AND CARE III APPLICABILITY

The CARE III program makes a major departure from many existing programs in that it was designed from the beginning to be a computer-aided engineering tool, and as such, incorporates the popular fault tree notation and structural fault-handling models with input parameters that can model many system redundancy management strategies. The CARE III mathematical model has been extensively scrutinized and documented. The program code was extensively tested and refined to be largely error free, and when errors do crop up (user or program generated), user-friendly aides or messages are presented.

The program is written in ANSI standard Fortran 77 code in single precision and will port to many different computers. The CARE III program can be executed either interactively or as a batch job. An interactive menu driven prompting program which aides the user in formatting the CARE III input data is available for execution on the Digital Equipment Corporation VAX-11/700 series computers under the VMS operating system.

CARE III predicts the unreliability of highly reliable reconfigurable fault-tolerant systems that include multiple digital computers or computer systems. A key feature of the CARE III capability is its ability to model very large systems that incorporate some form of system redundancy management strategy which controls hardware/software resources in the presence of multiple faults/errors of various types, i.e., permanent, transient, intermittent hardware faults/software errors.[1] The model in CARE III that accounts for the

---

1. The definitions of fault, error, and failure as applied to fault-tolerant systems do not have an industry standard interpretation. Even when a concerted attempt is made to use these terms consistently within a given document, exceptions crop up in their meaning because of their long term usage. Unfortunately, that situation applies to this document as well. The following definitions have been carefully applied throughout this document and should be interpreted as their first meaning; where historical usage mandates, the authors will acquiesce to their terminology, e.g., failure rate in lieu of fault rate.

   Fault - A condition which temporarily or permanently affects the ability of a module to perform its function.

   Error - A condition in which a module is incorrectly performing its function.

   Failure - Loss of system function.

system strategy of handling faults is the fault-handling model discussed in section 2.2 and appendix C.

System architectural characteristics that are modeled by CARE III include temporal and spatial redundancy to gain fault tolerance. Temporal redundancy is often used to recover from errors (e.g., transient errors) by "roll-ahead" or "roll-back" techniques. The effects of these strategies can be accounted for in the fault-handling model. Spatial redundancy is accomplished by using hardware redundancy.

Spatial redundancy is modeled as replicated modules where each module is assigned a failure distribution (hazard rate). What the analyst defines as a module is dependent on the level of the reliability assessment. Two factors should be considered when defining a module: a hazard rate must be assignable, and to utilize the fault-handling model, the system must have the capability of manipulating modules.[2] Manipulation involves module fault detection, module fault identification, and module reconfiguration. Reconfiguration can mean electrical isolation (a powered-down module) or it could mean signal isolation (module communication is ignored by other modules). When a set of exchangeable modules is assigned an identical hazard rate, the set is called a stage. Stages in turn, comprise a system. CARE III presently allows a fault-tolerant system to be constructed of one to seventy stages. The manner in which a system reconfigures module failures in a stage in order to tolerate faults is governed by an M(x) of N(x) specification and the fault-handling model. M(x) is the minimum number of modules required for stage x to function and N(x) the number of beginning modules in stage x prior to any failures.

To illustrate the M(x) of N(x) specification, consider a computer system stage composed of three identical computers that execute identical code using identical input data. The three computers all send outputs to a force-sum voter that actuates an aircraft control surface. To maintain aircraft control, two of the three computers must output correct data to the force-sum voter;

---

2. The first factor applies to every module. The second factor applies only to modules which are manipulated by the system redundancy management strategy. When modules are not manipulated by the system redundancy management strategy, as is the case of a hardware masking voter, the fault-handling model is not utilized by the user (default parameters are selected).

therefore, M=2 and N=3 for the computer stage. Each computer is a module and is assigned a hazard rate to model stochastic computer failures (fault occurrences). The hazard rate which is used to describe module failure probability is given by the Weibull hazard rate function which reduces to the common constant failure rate with the proper parameter setting, if desired. The user may assign a different hazard rate for each type of fault (permanent, transient, intermittent) that may afflict the computers.

When a fault occurs in a stage, the fate of the stage first depends on the action of the system redundancy management strategy which is emulated by the fault-handling model. If the fault is properly handled, the faulty module (e.g., a computer in the above example) is reconfigured and since two modules still survive, the system survives. If the fault is improperly handled, the stage may fail as well as the system. The stage will eventually fail given enough time because not enough modules will be left to guarantee fault free operation even though all previous faults may have been handled properly. Thus CARE III computes system unreliability in two parts: system unreliability due to imperfect fault-handling and system unreliability due to module depletion.

Many fault-tolerant systems are composed of several different sets of redundant modules with different hazard rates. These modules are partitioned into stages. System failure is then a function of combinations of stage failures. The failure combinations are described by a fault tree where the lower level events are stage failures, and stage failures are defined by M of N specifications (by defining input parameters M(x) and N(x)). The fault tree top event is system failure due to module depletion. Appendix G.1 depicts system trees for a number of aircraft flight control system designs. Stages may also be redundant. An example of stage redundancy is seen in example 8 of appendix G.2. The system tree shows the digital computer stage feeding into an AND gate with an analog computer stage feeding into the same AND gate through an OR gate with its switch. The analog computer stage is redundant with respect to the digital computer stage.

Most fault-tolerant systems are designed to tolerate all single faults. They are not designed, however, to tolerate all double faults. This tack is often taken to minimize hardware resources without incurring unacceptable system reliability. The system often tolerates many double faults without serious

consequences, e.g., two faults occurring in two different stages that are not dependent (not critically coupled). The redundancy management strategy simply configures out the failed modules one at a time. There is a class of double faults, however, which can be fatal to the system. These faults are called critically coupled faults, and they may occur in a stage or across two stages. An example of stage coupled faults occurs when two faults, one each in two of three different voting computers performs the same flight-crucial control computations. The two faults are not necessarily simultaneous occurring but most often one is latent (undetected) when the second occurs. The latent fault, although undetected by the system, may be generating numerous errors. When critically coupled faults occur, as specified by the analyst, CARE III considers this event catastrophic to the entire system and not solely to a stage or between two stages. The critical-pair specification is described by a special fault tree called a critical-pair tree. More detail can be found in section 3.6.

A popular system architecture utilizes a triad for fault masking and a number of spare modules for additional fault tolerance. CARE III treats these types of spare modules differently from redundant modules within stages (in-use modules). Spare modules can never incur a critically coupled failure with another spare or in-use module. Only in-use modules within or across stages can experience critical-pair failures. When spares are used with a masking voter, CARE III assumes that spare modules are handled by the system redundancy management strategy in the same manner as in-use modules, i.e., the same stage fault-handling model is used by both in-use and spare modules. One system redundancy management strategy that correlates with this assumption is one in which spares are "flexed" by continually retiring an in-use module (to spare status) while replacing that module with a spare (now an in-use module). The concept of spare flexing is employed to minimize latent faults by bringing spare units into active voting triads. The spare modules are always powered-up (hot spares). From a modeling point of view, spares flexing is equivalent to off-line spares self-testing. In this regard, CARE III models architectures that employ off-line self-testing. The specification that prescribes modules spare status is the NOP (number operational) array and is discussed in detail in section 3.2. Further discussion on module status is covered in appendix B. These details are of particular interest to users who wish to model module dependency.

The CARE III model and program have received extensive scrutiny by several independent researchers and engineering users. The mathematical model was derived independently by the Boeing Computer Services Company in two ways: through available documentation and from examining the computer code. The program was applied to numerous test cases and practical engineering system designs. Although the absence of program and modeling errors are not guaranteed by this extensive scrutiny, those that may appear should be rare. Confirmed program errors should be brought to the attention of the CARE III project engineer.

The increased sophistication and complexity of fault-tolerant systems has brought about the requirement for a reliability assessment tool for these systems. Unfortunately, the reliability tool required to fill the need must also be more sophisticated and complex than traditional tools. CARE III, which is such a tool, was designed to minimize user interface complexity; however, the user must have a good understanding of the CARE III model in order to properly use this tool. Section 2.0 is provided as a brief introduction to the CARE III model. More information can be obtained from references 1, 2, 4, 5, and 6.

2.0 CARE III MODEL - HIGH LEVEL VIEW

As outlined in the introduction, the user interacts with CARE III via a system tree specification, one or more critical-pair tree specifications if required, and one or more fault-handling models. The details of the mathematical model that integrate the fault tree models and the fault-handling model can be found in numerous documents, but are not required to use CARE III as an engineering tool. Some exposure to the high level math model is required, however, so that the analyst can properly interpret the CARE III output data.

2.1 GENERAL MODEL STRUCTURE

Figure 2-1 shows the general structure of the high level (aggregate) CARE III model. Each ellipse represents a state of a system and is partitioned into the $G(\underline{\ell})$, $H(\underline{\ell})$, and $F(\underline{\ell})$ states where $\underline{\ell}$ is a vector,

$$\underline{\ell} = (\ell(1), \ell(2), \ldots, \ell(x))$$

with $\ell(x)$ elements so that $\ell(x)$ is the number of faulty modules[3] within stage x and

$G(\underline{\ell})$ represent system operational states which may contain faults

$H(\underline{\ell})$ represent module depletion states which cause system failure

$F(\underline{\ell})$ represent system failure states due to imperfect fault handling

The system enters one of the H failure states when a combination of stage failures occurs which leads to the system fault tree's top event (see system fault tree section 3.5). Stage failures occur when the number of faulty modules, $\ell(x)$, for each failed stage exceeds that stage's user defined limit of $N(x) - M(x)$ modules. F failure states are entered when the system is unable to cope with faults/errors.

---

3. In CARE III permanent and intermittent faults are handled in a different manner than transient faults. Transient faults increment the fault variable $\ell(x)$ only when the module is detected as faulty and permanently isolated from the system. Non-transient faults, permanent and intermittent, increment the fault variable $\ell(x)$ when they occur. Thus $\ell(x)$ is the number of faulty non-transient modules plus number of isolated modules with transient faults.

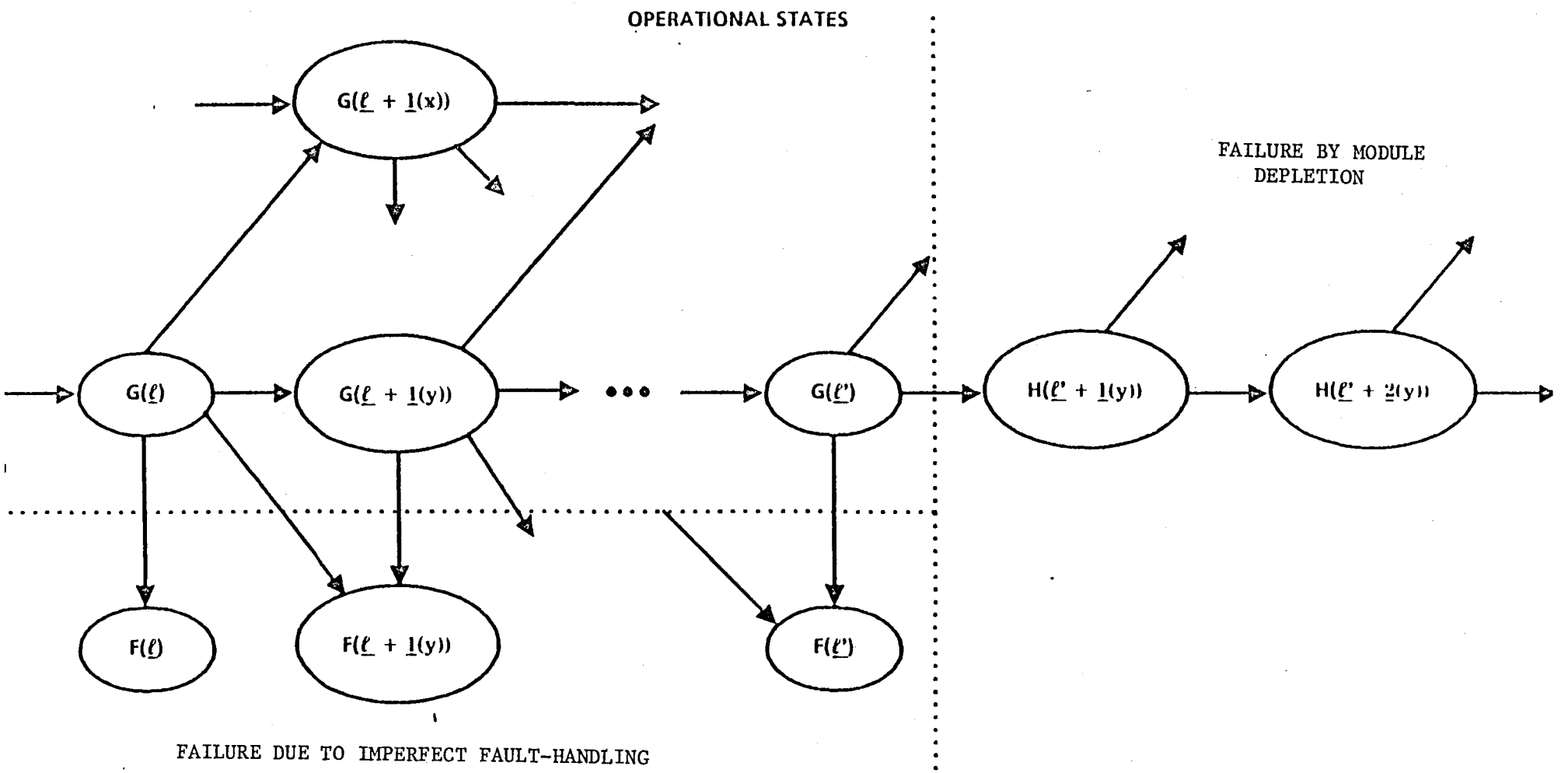


Figure 2-1 General Structure of CARE III Aggregate Model

The following probability definitions[4] are important to the user as they are CARE III outputs:

$P(t|\underline{\ell})$ probability of the system being in state $G(\underline{\ell})$ at time t

$P^*(t|\underline{\ell})$ probability of the system being in state $H(\underline{\ell})$ at time t

$Q(t|\underline{\ell})$ probability of the system being in state $F(\underline{\ell})$ at time t

Further interpretation of figure 2-1 follows. A transition from a $G(\underline{\ell})$ to an $F(\underline{\ell})$ state may represent a single fault system failure (when it is defined by the user). A transition from state $G(\underline{\ell})$ to $G(\underline{\ell} + \underline{1}(y))$ represents a successful system recovery from the occurrence of a fault in stage y. $G(\underline{\ell} + \underline{1}(y))$ represents an operating system configuration with one more fault in stage y than $G(\underline{\ell})$ has. A transition from $G(\underline{\ell})$ to $F(\underline{\ell} + \underline{1}(y))$ represents a coexisting undetected pair of faults that are critically coupled in the system which cause system failure.[5] Transitions of this type exist when the user defines a critical-pair tree (see section 3.6).

Of particular importance are the values P*SUM and QSUM, i.e., $\Sigma\ P^*(t|\underline{\ell})$ where the sum is over all H states and $\Sigma\ Q(t|\underline{\ell})$ where the sum is over all F states. P*SUM is the probability of the system being in any H state at time t, and QSUM is the probability of the system being in any F state at time t. P*SUM + QSUM is the system unreliability, the final desired result.

Veteran users of reliability assessment models have developed a healthy respect for caution in applying and interpreting model results. Every model is an abstraction, hence, an approximation to some physical system. Therefore, the user must pay close attention to the modeling assumptions and approximations built into the modeling methodology. The most important ones for CARE III are summarized in appendix D; greater detail can be obtained from references 2, 4, 5, and 6.

---

4. $P^*(t|\underline{\ell})$ values are not listed by CARE III, only P*SUM is.

5. The definitions of state transitions as described here are an over simplification. Other factors are involved in causing the transitions that also depend on whether or not a transient model is defined. Reference 6 provides this detail.

Although CARE III was designed to model a very large class of system designs, undoubtedly not every system design will be directly applicable for use by CARE III. In some cases, it may be necessary for the user to do some pre/post processing to accommodate such system designs. An example of this necessity involves the modeling of critical-triple failures, i.e., a system possessing three coexisting faults located in modules performing critical functions. Post processing involving a convolution integral may be required in this case.

2.2 FAULT-HANDLING MODEL

The fault-handling model describes how a module fault/error of a specified type behaves and is handled by the system. The fault-handling model allows three types of fault-handling behavior to be modeled: permanent, intermittent, and transient. A permanent fault has the property that once a defect occurs in a module, it remains indefinitely and maintains only one manifestation. The module experiencing an intermittent fault has incurred a permanent defect but exhibits two manifestations: faulty or nonfaulty behavior. When exhibiting its faulty behavior, the module behaves like a permanent fault. During its nonfaulty (benign) mode, the module is operating properly. A transient fault is not caused by a permanent defect within a module, but rather manifests a faulty behavior for some finite time and then the module is fault free. The module may experience a subsequent transient or other fault type.

Figure 2-2 shows the general single fault-handling model which is a composite model representing the three types of faults. The different fault types are modeled by assigning appropriate values or functions which connect the circular model states. One set of connecting values or functions (transition parameters) defines a single fault-handling model, i.e., the handling of a fault type. CARE III allows the user to identify a maximum of five specific fault-handling models in this way.

The states in the fault-handling model are given below. The parameters governing transition among these states are shown in figure 2-2 and are defined in section 3.1. It is important to note that the interpretation given to the model states and state transition parameters are subjective. One such interpretation is given in this report. The reader is encouraged to be inventive in using these models and not accept the interpretation given here as

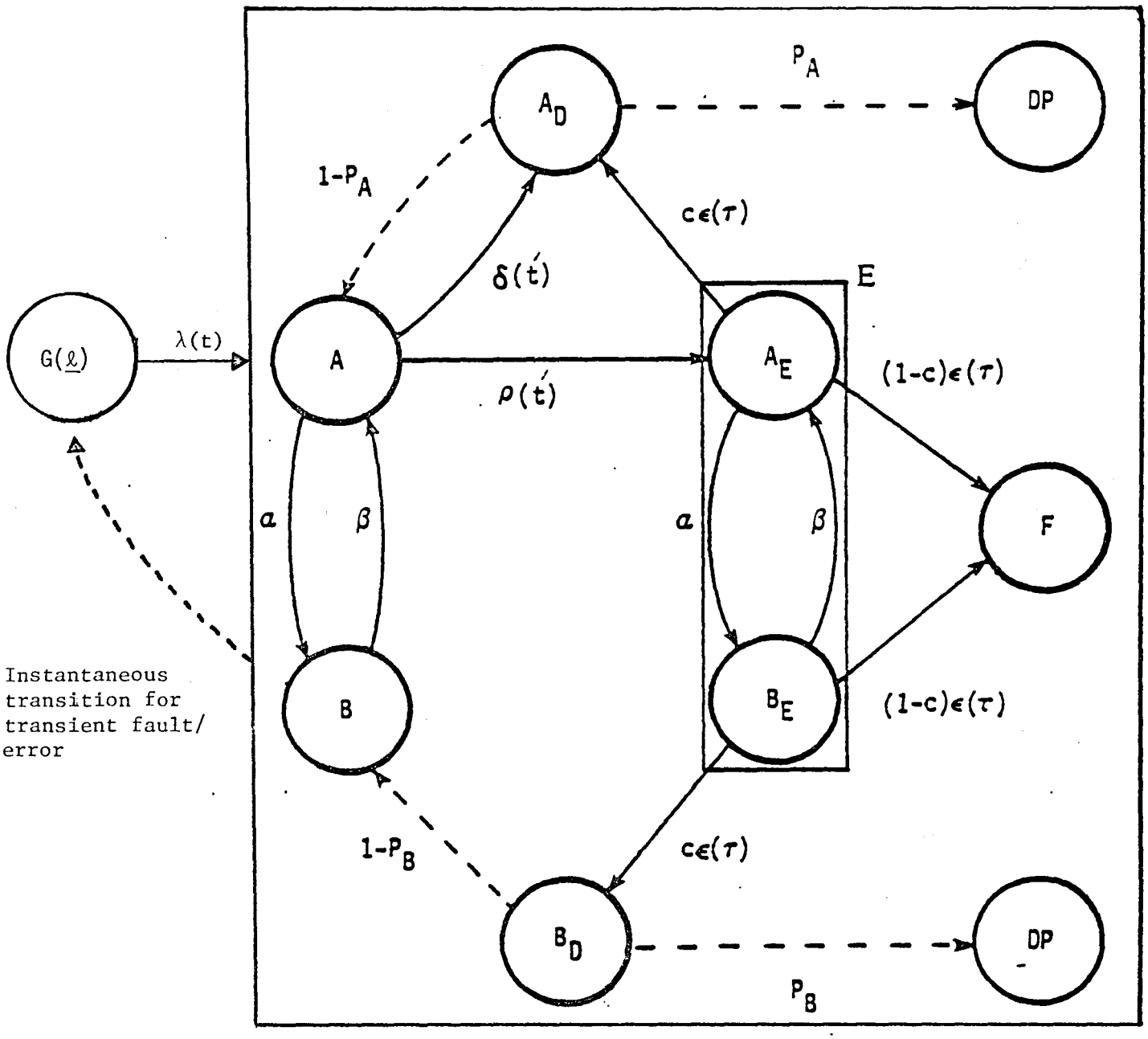
the only one; however, the values of the parameters used must be reasonable or else the CARE III program may not execute correctly (see appendix D for caveats).

A ACTIVE FAULT - Latent (undetected) fault state with no errors. A module has experienced a fault but is performing its functions correctly. The system is unaware of the presence of the fault. Entry to this module state, given by $\lambda(t)$, could result from a number of possible causes. For each stage, the user may define up to five distinct fault occurrences, $\lambda(t)$, in the form of Weibull or constant failure rates (see section 3.3, for OMG(i,x) and RLM(i,x)). The fault occurrences could be attributed to but not limited to the following sources: permanent, transient, intermittent, hardware, and/or software anomalies. Since the fault is active, the system can excite the fault to produce errors, i.e., intentionally by executing a self-test program or unintentionally by executing application code, resulting in a transition to state A_D or A_E.

A_D ACTIVE DETECTED - Detected fault/error state. A module has been detected as faulty, i.e., the fault or error(s) was detected. Transition to this state can occur in one of two ways: A $\delta(t)$ transition can be interpreted as resulting from a protected machine condition in that the system was executing a self-test/recovery program. The other path via $\rho(t)$ and $C\varepsilon(t)$ can be interpreted as an unprotected machine condition in that the fault produced an error(s) while an application program was being executed and was detected by a majority voter or comparison-monitoring algorithm. The unprotected machine condition is potentially more dangerous than protected because it is expected that fault detection in the unprotected case may take longer allowing a greater possibility for a second critically-paired fault (if defined) to occur which is catastrophic to the system.

A_E ACTIVE FAULT-ERROR - Latent fault and error state. A module is incorrectly performing a function or operation. The fault was excited by application code to produce an error(s). The system is unaware of the presence of an error(s) or the fault that produced the error(s).

DP DETECTED AS PERMANENT - Reconfiguration state. A module is permanently isolated from the system.



A: ACTIVE
B: BENIGN
D: DETECTED
E: ERROR
F: FAILURE
DP: DETECTED AS PERMANENT (NON-TRANSIENT)

t' = time from entry into active state A

τ = time from entry into error state E

t = operational time

Figure 2-2. - General Single Fault-Handling Model (shown in box).

F FAILURE - System failure has occurred due to inability to handle a single fault shown in figure 2-2.

B BENIGN FAULT - Benign fault state, no errors. A module is in a benign state, and as long as the module is in this state, it will not be detected as potentially faulty because no error exists in the module. For a transient fault, there is an instantaneous transition to the originating operational state $G(\underline{\ell})$.

B_E BENIGN FAULT LATENT ERROR - Benign fault, latent error state. A module is in a benign state after incorrectly performing a function or operation. The source of the current error(s), the fault, is no longer producing new errors; however, at least one error is still in the system, and it is undetected.

B_D BENIGN FAULT, DETECTED ERROR - Benign fault, detected error state. A benign module has been detected as faulty as a result of the error it produced even though the cause of the error, the fault, has vanished.

For a permanent fault, α and β are zero (see section 3.1 for the parameter definition). The affected module when called upon to perform a function will always perform that function erroneously. There may be a delay between when the fault has occurred, state A, and when the module begins to generate errors, state A_E (figure 2-2).

For an intermittent fault, both α and β are positive. The affected module will only intermittently perform its function incorrectly. CARE III views an intermittent as a physically defective module. If an intermittent is correctly diagnosed, the correct action should be to purge the faulty module.

For a transient fault, α is positive but β equals zero. The module may perform erroneously only during the "short passage" of a transient (e.g., power surge) through the system. If the module passes to the benign state B, it instantaneously returns to a fault-free state where it may incur new faults, transient or other types. This unique property of a transient fault differentiates it from non-transients and requires special treatment in the reliability model (see footnote 3, section 2.1). Since a module having a

transient is not viewed as a physically defective module, purging a module for a detected transient is often considered poor utilization of hardware resources. Choosing the proper values for P_A and P_B (see section 3.1) gives the user the capability of modeling the fate of a transient appropriately.

When the single-fault model is specified for a stage, it represents the fault handling of all modules in that stage which includes both in-use and spare modules (see appendix B.1). Since fault-handling parameters will likely differ between in-use and spare modules in most applications, the analyst should use conservative values. The CARE III model allows a stage to be subject to a maximum of five different kinds of fault occurrences at one time. For example, a central processing unit (cpu) in a computer might be subject to permanent stuck-at-one logic faults occurring at a constant rate of 2.0E-5 faults per hour and permanent stuck-at-zero logic faults occurring at 3.5E-5 faults per hour. These two kinds of faults may both be linked to the same permanent fault-handling model created by the user (the redundancy management strategy will handle the two faults in the same way) or they may be linked to separate fault-handling models. For this example, note that the cpu is subject to a total of 5.5E-5 faults per hour.

Every kind of fault identified for a CARE III application must be assigned its own hazard rate. For the maximum number of stages (70), up to 350 hazard rates could be assigned. Each kind of fault for each stage must then be linked to one of the five (or less) fault-handling models created by the user. If one had a model using the maximum number of stages (70) and the maximum number of identifiable faults, some identified kinds of faults would have to use one or more of the fault-handling models more than once.

In addition to single fault system failures that are described by the single fault-handling model, CARE III also can model double fault system failures. (See figure C-1 for an illustration of double fault handling.) Two modules which have incurred faults that are critically paired can cause system failure before the system can detect, isolate, and replace the faulty modules. The motivation for including a critical pair fault model is based on the concept that there is a class of synergistic paired faults which has the capability of causing the loss of a system critical function. The two faults are not simultaneous occurring faults but rather coexisting latent (undetected) faults.

The fact that the faults are latent doesn't mean they are dormant or benign, but on the contrary they are treated as jointly active and devastating. Usually the exact nature of the synergism is difficult to assess and, in fact, specific cases may be cited where the synergism breaks down. Often to cope with the modeling complexity, the conservative approach is taken; allowing the critical-pair synergism to exist irrespective of the few exceptional cases. Modules which are critically paired may be from the same or two different stages. An example of a stage critical-pair failure is the occurrence of two faults, one each in two of three different voted computers performing the same flight-crucial control computations. A fault in a spare computer and a fault in the voting triad, however, would not be critically coupled. An example of a critical-pair failure across stages would be a critically coupled fault between a computer module and a computer bus module as illustrated in appendix G.2 example problem 7. Since three voting computers communicate over three buses, a fault in a voting triad and a coexisting fault in a bus not connected to the faulted computer would preclude a correct majority vote at the receiving end of the triplex bus.

The critical-pairs information is defined by a CRITICAL-PAIR TREE, section 3.6. The sparing information specified by NOP, section 3.2, has an important effect on the evaluation of double-fault failures. Figures 2-2 and C-1 illustrate the complete general models. By judicious selection of parametric values including defaults, a number of specific models may be defined. Details of the double-fault model, illustrated by figure C-1, are covered in appendix C.

3.0 INPUT

A simple fault-tolerant system and the CARE III input data are defined below to portray the general input structure. Figure 3-1 delineates the input data in the NAMELIST format while figure 3-2 delineates it in the list-directed format (see appendix E for details). Figure 3-1 shows the data divided into six paragraphs: FLTTYP, STAGES, FLTCAT, RNTIME, SYSTEM FAULT TREE, AND CRITICAL-PAIR FAULT TREE. After describing the example problem, each paragraph will be discussed in relation to the example and will further be generalized to cover the complete capability.

An example input will now be provided which illustrates the most important input statements. Consider a system composed of three stages: #1 - processor; #2 - memory; #3 - power supply. There are six processors, two active triads with no spares; five memories, one active triad with two spare modules; and two power supplies, one in-use and one spare. The system fails by depletion of modules if any of the three stages has failed. The minimum number of operational modules required is two for the processor stage, two for the memory stage, and one for the power supply. When a power supply fails, power is immediately supplied by the spare (perfect fault handling). For the processor, a system failure can occur if there are two faulty modules that are critically coupled within a triad (imperfect fault handling). Similarly, two faulty memory modules that are critically coupled in the triad can cause system failure. The processor is susceptible to permanent faults with a constant rate of 2.2E-4 per hour; the memory is susceptible to permanent faults with constant rate 3.5E-5; and the power supply is susceptible to permanent faults with constant rate of 1.5E-6. The design information assumes that a single fault in the processors or memories cannot cause system failure. Processors and memories are governed by the same system fault-handling process. Average time for self-test identification, isolation and replacement of a faulty module is 4.06 seconds; average time for a faulty module to generate errors is 188.5 msec.; average time for detection, isolation, and replacement of a module generating errors is 342.9 msec. All rates are assumed to be constant. The input to this problem is shown in figures 3-1 and 3-2 for both NAMELIST and list-directed formats.

A detailed description of the input data and format are discussed in sections 3.1 - 3.6. A summary of the input paragraphs providing input ranges and default values is given in section 3.7. These paragraphs must be input in the order described in sections 3.1 - 3.6. The first four paragraphs must always be provided. The next two paragraphs are required if flags in the RNTIME paragraph are set .TRUE.. These paragraphs describe fault trees. The format for the first four paragraphs is given in the NAMELIST format for VAX-11 Fortran and CDC Fortran V compilation and in list-directed format for general Fortran 77 compilation. The last paragraphs describing fault trees have the same syntax in both cases.

For the NAMELIST input, information begins in column two; column one is ignored. Each paragraph begins with $XXX where XXX is the paragraph identifier. Variables defined in a paragraph may appear in any order. Commas separate input lines. Each paragraph is terminated by a $. The format for FAULT TREE inputs is addressed in sections 3.5 and 3.6. The user is not required to specify every NAMELIST input value. Input parameters are preset with default values by CARE III. If a value is not changed by the user, a default value will be used. The default values are summarized in section 3.7. Section 3.8 discusses the availability of some data input values.

For list-directed input, information begins in column one. List-directed input does not use identifiers for input parameters. Also, unlike NAMELIST input, the order in which input parameters are listed for list-directed input is very important. The required order is given in section 3.7 and is discussed in appendix E. If a default value is desired, it must be indicated by including the comma delimiter to fill its place.

```
$FLTTYP     NFTYPS          = 2,
            MARKOV          = 1,
            DEL(1)          = 8.87E2,
            RHO(1)          = 1.91E4,
            C(1)            = 1.0,
            EPS(1)          = 1.05E4$
$STAGES     NSTGES          = 3,
            N(1)            = 6,
            M(1)            = 2,
            NOP(1,1)        = 6,
            NOP(2,1)        = 3,
            N(2)            = 5,
            M(2)            = 2,
            NOP(1,2)        = 3,
            N(3)            = 2,
            M(3)            = 1,
            IRLPCD          = 3$
 $FLTCAT    NFCATS(1)       = 1,
            NFCATS(2)       = 1,
            NFCATS(3)       = 1,
            JTYP(1,1)       = 1,
            RLM(1,1)        = 2.2E-4,
            JTYP(1,2)       = 1,
            RLM(1,2)        = 3.5E-5,
            JTYP(1,3)       = 2,
            RLM(1,3)        = 1.5E-6$
 $RNTIME    FT              = 10.0,
            SYSFLG          = .TRUE.,
            CPLFLG          = .TRUE.$
SYSTEM FAULT TREE FOR THREE STAGES IN SERIES
1 3 4 4
4 0 1 2 3
CRITICAL-PAIR FAULT TREE
1 13 20 23
1 1 6
2 7 11
3 12 13
20 2 1 2 3
21 2 4 5 6
22 2 7 8 9
23 0 20 21 22
```

Figure 3-1 CARE III input file using NAMELIST syntax
(Order of inputs within a paragraph is unimportant)

```
2,          ,          ,
            ,          ,
     8.87E2,           ,
     1.91E4,           ,
     1.05E4,           ,
            ,          ,
            ,          ,
            ,          ,
            ,          ,
            ,          ,
            ,          ,
        1.0,           ,
     , , , , ,1, /
3,   6, 5, 2,
     2, 2, 1,
     6, 3,  ,  ,  ,
     3,  ,  ,  ,  ,
      ,  ,  ,  ,  ,
      ,  ,  ,
     3, , /
1, 1, 1,
       1,       1,       2,
       ,        ,        ,
     2.2E-4, 3.5E-5, 1.5E-6/
10.0, , , .TRUE., .TRUE., , , , , , , /
SYSTEM FAULT TREE FOR THREE STAGES IN SERIES
1 3 4 4
4 0 1 2 3
CRITICAL-PAIR FAULT TREE
1 13 20 23
1 1 6
2 7 11
3 12 13
20 2 1 2 3
21 2 4 5 6
22 2 7 8 9
23 0 20 21 22
```

Figure 3-2 CARE III input file using list-directed syntax
(Order of inputs within a paragraph correspond to section 3.7)

3.1 FAULT HANDLING

NAMELIST syntax:

```
$FLTTYP     NFTYPS      = 2,
            MARKOV      = 1,
            DEL(1)      = 8.87E2,
            RHO(1)      = 1.91E4,
            C(1)        = 1.0,
            EPS(1)      = 1.05E4$
```

List-directed syntax:

```
2,          ,           ,
            ,           ,
        8,87E2,         ,
        1.91E4,         ,
        1.05E4,         ,
               ,        ,
               ,        ,
               ,        ,
               ,        ,
               ,        ,
           1.0,         ,
, , , , ,1, /
```

The fault-handling paragraph defines how the system handles faults that have occurred. The user assigns values to fault-handling parameters listed in the paragraph which define characteristics of the fault type (permanent, intermittent, or transient) and how the system handles/responds to the fault. Fault type characteristics are in turn modeled by rate functions[6] which "connect" the different states of the single and double fault-handling models. Transitions between some states occur instantaneously. Probability values are used for these types of transitions (see figs. 2-2 and C-1).

Up to five fault-handling models may be defined for up to five different fault types. It is also possible to have five fault-handling models for one fault type, e.g., five different transient-handling models. A fault-handling model is defined by one set of fault-handling parameters for the single-fault model shown in figure 2-2. The fault-handling model(s) may be assigned (described later) to any defined stage, e.g., one to five models for a given stage or they can be distributed across all stages in any combination. By assigning appropriate values to the fault-handling parameters, transition rates are defined which permit the formation of many variations of this model. Although every transition rate must be defined by the user or by default, transitions can be precluded by assigning very slow rates. Alternately, transitions can be made to appear instantaneous by assigning very fast rates. Some caution must be exercised to maintain a reasonable separation of fault occurrence and fault-handling rates, however (see section 3.7 and appendix D).

---

6. Rate functions are derived from user selected probability density functions.

In the example problem, two fault-handling models are required, one for the processor and memory stages and a second for the power supply stage, so NFTYPS = 2. The parameters listed in this paragraph describe the fault-handling model for the processor and memory stages. Since it is assumed that the power supply stage handles its faults perfectly, a fault-handling model different from the processor memory stages must be assigned to the power supply stage. Ordinarily the power supply stage will require non-defaulted parameters to properly model its fault handling; however, here it is assumed that fault handling is accomplished perfectly. Therefore, a default fault-handling model is selected by setting NFTYPS = 2 and by not including parameter values for the second fault-handling model in this paragraph.

$FLTTYP fault-handling input identifier which must appear at the start of the paragraph. If the user is using list-directed formatting to create his input file, he should note the parameters discussed below are not discussed in the order required by list-directed format. The user must read section 3.7 and appendix E.2 for the format requirements if using the list-directed format.

NFTYPS number of fault models to be included. The same fault type may be identified with several stages. In doing this, the analyst is stating that the system responds to a fault in the same manner for each stage, although the stages may have different fault-occurrence rates. The value of NFTYPS must be at least one, even if there are no single fault failures or no critical-pair faults in the system. CARE III requires that at least one fault-handling model be assigned to each stage even though some stages may have perfect fault-handling. For example, if a system has stages with all perfect fault-handling, then NFTYPS = 1. If the system has one stage with perfect fault-handling and all the others are imperfect and modeled by one fault model (as in the example), then NFTYPS = 2.

MARKOV variable which defines whether all fault-handling models have exponential distributions (constant transition rates) only, and hence the fault-handling model is homogeneous Markovian. The numerical procedure used for the homogeneous Markovian model, MARKOV = 1, is numerically stable and very efficient. If the general fault-handling

model (semi-Markov) is used, the numerical procedure implemented is much slower and may have stability problems if parameter values in the fault-handling model are separated by several orders of magnitude. Recent studies have indicated that the substitution of exponential distributions for uniform distributions produces minute differences in many cases and suggests the use of the Markov fault-handling model at least for preliminary analysis.

1 - use homogenous Markov solution. All transition times are exponential (constant rates) irrespective of user assigned input distributions.

2 - use general solution. At least one transition has a uniform distribution. MARKOV must be set to 2 in order to get plots of fault-handling intensity functions (see CVPLOT in section 3.1; and appendix A.2)

The rate functions $\delta(.)$, $\rho(.)$, and $\epsilon(.)$ shown in figures 2-2 and C-1 are of the form:

$f(t) = \theta \qquad 0 \leq t \qquad$ for exponential t

or

$f(t) = \theta/(1-\theta t) \qquad 0 \leq t \leq 1/\theta \qquad$ for uniform t

where the user specifies the rate functions form (exponential or uniform) and values for θ. θ values are specified in the input by the array names, DEL(i), RHO(i), and EPS(i) for fault type i. The reader should note that only when the exponential rate function is selected will a rate function be equal to its corresponding array value, e.g., $\delta(t) = f(t) = \theta$ = DEL(i) for exponential t. ALP(i) and BET(i) are always exponential rate functions. Rate functions are always specified in the units, events/hour irrespective of ITBASE selection (see section 3.4). For an exponential distribution, an event occurs with a mean time of $1/\theta$ hours. For a uniform distribution, an event occurs with a mean time of $1/(2\theta)$ hours. An exponential distribution (constant detection rate) may be used when a diagnostic program (self-test program) is randomly scheduled for execution. In this case, detection is most likely to occur early in time, but allows for non-detection for long operational times. The uniform distribution may be used to approximate the effects of executing the diagnostic program on a fixed scheduled. In this regard, uniform detection means that detection will occur within the user specified time ($1/\theta$), but can occur equally likely during that time.

The presence of any of the following parameters in the $FLTTYP paragraph is used to define the fault-handling model. The absence of one or more parameters causes CARE III to select default values (see section 3.7).

DEL(i) parameter for describing the rate function between active fault state A to detected state A_D. DEL(i) has the units, events/hour (see end of table for more definition).

RHO(i) parameter for describing the rate function from the active fault state A to the active fault-error (erroneous operation) state A_E. The expression using this parameter is used to model the rate at which a fault generates errors. RHO(i) has the units, event/hour.

EPS(i) parameter for describing the rate function from error state to detected state A_D, B_D or to the single-fault system failure state F. EPS(i) has the units, events/hour (see end of section 3.1 for more definition).

IDELF(i) indicator variable[7] defining if DEL parameter is used for an exponential or uniform rate function. IDELF values are disregarded if parameter MARKOV = 1.

1 - exponential rate function
2 - uniform rate function

IRHOF(i) indicator variable defining if RHO parameter is used for an exponential or uniform rate function. IRHOF values are disregarded if parameter MARKOV = 1.

1 - exponential rate function
2 - uniform rate function

IEPSF(i) indicator variable defining if EPS parameter is used for an exponential or uniform rate function. IEPSF values are disregarded if parameter MARKOV = 1.

1 - exponential rate function

---

7. For the example problem of this section, the CARE III defaults for parameters ALP, BET, IDELF, IRHOF, IEPSF, PA, PB, DBLDF, TRUNC, CVPRNT, CVPLOT, IAXSCV, and LGTMST will be used since these parameters are not set in the input file.

2 - uniform rate function

PA(i) probability that a module detected as faulty (a fault or error was detected) in the active detected state A_D is isolated from the system. The default value of 1.0 implies that all modules identified as faulty are isolated from the system. Setting this parameter to a value less than unity allows the analyst to model system redundancy management schemes where the management scheme has to decide if an isolated fault was either a transient or non-transient fault. For non-transient faults the value of PA will normally be a value close to one. For transient faults the value of PA will be close to zero.

PB(i) probability that a module detected as faulty (an error was detected) in the benign fault detected error state B_D is isolated from the system (for intermittent or transient faults only). The default value of 0.0 implies that if a module error is detected while in the B_D state, the module is not deleted from the system, but presumed, possibly from additional tests, to be fully operational. For a transient fault, this would be the correct decision.

C(i) probability that a faulty module in the A_E or B_E state is not lethal to the system. This probability will only be taken into account when both of the parameters RHO and EPS are greater than zero. This parameter provides the means for modeling system failure due to a single fault.

The next two parameters determine which of three possible fault types is being described. If ALP(i) and BET(i) both equal zero (the default value), the ith fault-handling model type is a permanent one. If ALP(i) and BET(i) are both greater than zero, an intermittent fault-handling model is being described. Finally, if ALP(i) is greater than zero and BET(i) equals zero, a transient fault-handling model is described.

ALP(i) parameter for constant rate transition from active state A or A_E to benign stage B or B_E. ALP(i) has the units, events/hour.

BET(i) parameter for constant rate transition from benign state B or B_E to active state A or A_E. BET(i) has the units, events/hour.

Some additional discussion on the interpretation of the transition parameters is required in order for the user to properly use the parameters. The physical interpretation of $\delta(t)$ is manifold and depends on the fault type. As an example, if the user is only modeling non-transient faults, $\delta(t)$ is a composite function that models the time to detect the fault, identify the faulty module, and reconfigure the module from the system. Because only non-transient faults are being modeled, parameter PA will be set to unity (default value) guaranteeing module isolation from the system.

If the user is modeling a transient, he has several choices in defining the meaning of $\delta(t)$. One redundancy management strategy is to treat transients like non-transients as described above. Another strategy is to wait some small time duration to allow fast transients to vanish. In a known heavy transient environment, the first strategy would waste modules; however, the probability of incurring a second fault while the first is being handled (a critically coupled system failure) is significantly reduced. The latter strategy of waiting for a transient to vanish may better conserve hardware resources but brings with it a greater probability of incurring a critically coupled system failure. By setting PA to a value of less than unity, the user can model the latter strategy. 1.0 - PA is the probability of not reconfiguring a known faulty module from the system, i.e., return to state A. The parameter $\delta(t)$ takes on a different meaning than that for non-transients when PA < 1.0 because $\delta(t)$ for non-transients includes reconfiguration time while a module containing a transient fault may return to state A and thus not include reconfiguration time. The distinction between $\delta(t)$ for transients and non-transients may in practice be a moot point since the significant time factor contributing to $\delta(t)$ is often the fault detection time.

The interpretation of the $\varepsilon(\tau)$ transition parameter is intimately linked to the parameter C. If C = 1.0, then transition to state F for a single fault/error is precluded. In this case, $\varepsilon(\tau)$ can take on a similar interpretation to $\delta(t)$. The difference being that $\varepsilon(\tau)$ models elapsed time after an error occurs while $\delta(t)$ models elapsed time to state A_D when the detection mechanism is something other than comparison of outputs. The parameter $\rho(t)$ acts as a time delay which slows down the error recovery process and makes a system failure due to critical-pair faults more probable. When C = 1.0, the system will always recover from single faults/errors provided that there exist redundant modules

in the system stages. Setting C < 1.0 allows the possibility of system failure resulting from a single fault/error by transition from state A_E to F or by a single error by transition from state B_E to F.

If the general fault-handling model is used, MARKOV = 2, the following parameters will be used:

DBLDF parameter governing the step doubling rule used in the numerical integration of the general fault-handling model functions. Although the range allowed is from 0.01 to 0.1, a value of 0.01 is recommended for controlling accuracy. A value of as large as 0.05 will significantly reduce the fault-handling model computation time but may provide inaccurate answers if there is moderate disparity between transition rates.

TRUNC fault-handling function's truncation value. See appendix F for more detail.

In addition to the end product, system reliability, the user may output some of the fault-handling functions. For most reliability analyses, this output is of limited use and may be suppressed (the default option). The controls for the fault-handling outputs are CVPRNT and CVPLOT:

CVPRNT flag for outputting moments of single and double fault-handling functions. The data generated by setting this flag to .TRUE., is used by the program developers and is not normally of interest to the reliability analyst.

CVPLOT flag for plot of single and double fault-handling functions. Setting this flag to .TRUE. will cause the COVRGE module of the CARE III program to generate two plot files called SNGFL and DBLFL. These functions are:

1. $P_B(t)$ - probability single fault is benign
2. $P_{\bar{B}}(t)$ - probability single fault is not benign
3. $p_F(t)$ - single fault failure intensity
4. $P_L(t)$ - probability of latent single fault
5. $p_{DP}(t)$ - single fault detected as permanent intensity

6. $p_{DF}(t)$ - double fault failure intensity

This plotting capability was implemented for debugging purposes and may be of little interest to the user. See appendix A.2 for plotting options.

IAXSCV Y-axis scale for plotting fault-handling functions.

1 - linear Y axis
2 - log Y axis (default value)
3 - linear and log Y axis plots
4 - log-log plot only

LGTMST flag governing the choice of logarithmic or linear time steps used in the integration algorithm for computing the $P(t|\underline{\ell})$ and $Q(t|\underline{\ell})$ probabilities. The default value .TRUE. gives logarithmic time steps.

3.2 STAGE DEFINITION

NAMELIST syntax:

```
$STAGES     NSTGES      = 3,
            N(1)        = 6,
            M(1)        = 2,
            NOP(1,1)    = 6,
            NOP(2,1)    = 3,
            N(2)        = 5,
            M(2)        = 2,
            NOP(1,2)    = 3,
            N(3)        = 2,
            M(3)        = 1,
            IRLPCD      = 3$
```

List-directed syntax:

```
3,  6, 5, 2,
    2, 2, 1,
    6, 3,  ,  ,  ,
    3,  ,  ,  ,  ,
     ,  ,  ,  ,  ,
     ,  ,  ,
    3, , /
```

The stage definition paragraph defines the number of stages in the system, the number of modules within a stage, the minimum success status for a stage and the operational configurations for a stage. The type of output is also specified.

$STAGES stage identifier which must appear at the start of the paragraph.

NSTGES number of stages in the system.

N(x) number of identical modules in stage number x.

M(x) minimum number of modules needed for stage x to be operational.

NOP(j,x) specifies the jth operational configuration for stage x and therefore specifies which (in-use) modules are subject to critical-pair system failures ($1 \leq j \leq 5$). For a stage, this array specifies the allocation of modules to in-use and spares as modules are deleted from the system. Spares do not contribute to critical-pair system failures. Although spares do not contribute to critical-pair system failures, they can contribute single/faults causing system failure and to system unreliability if the parameter C for the fault-handling model for stage x is less than one. For each stage, up to five NOP values may be defined. If D(x) modules have been deleted, N(x)-D(x) modules are available. The number of in-use modules is the largest NOP(j,x) less than or equal to the number available; the remaining modules are spares. If N(X)-D(x) is less than the smallest NOP specified, but greater than or equal to M(x), the minimum number of units needed, then all N(x)-D(x) are in-use and there are no spares. If NOP is not specified for a stage, all non-deleted modules are assumed to be in-use. No modules are spares. Thus, when no NOP inputs are given for stage x, the default NOP array used is the decreasing sequence N(x), N(x)-1,..., M(x). The sparing architecture described by NOP affects only the double-fault error computations. If no critically coupled faults for any stages are specified through the critical-pairs fault tree, the NOP parameters have no effect and need not be specified.

In the example above, stage 1 starts out with six in-use modules and zero spares, designated by NOP(1,1) = 6. When one module has been deleted, five modules are available. With NOP(2,1) = 3, the largest NOP less than five is three. Therefore, after one module in stage one has been deleted, three modules are in-use and two modules are spares. If four modules are deleted, two modules are available. Because a NOP(3,1) value was not specified, the default of NOP(3,1) will be one less than the specified NOP(2,1) value. Thus, when four modules for stage one are deleted, two modules will be in-use and

there will be no spares. For the general case, the number of in-use modules reduces by steps of one, from the last specified NOP value, down to the value of M. See appendix B for more discussion on how NOP parameters are used.

LC(x) parameter[8] that inhibits some critical-pair failure computation of $Q(t|\underline{\ell})$ values with less than LC faults occurring in stage x. See appendix F for more details.

IRLPCD specifies the option chosen for output print-out. One may obtain summary results only, or for each fault vector, $\underline{\ell} = (\ell(1), \ell(2),..., \ell(x),...)$, the probability of system failure due to imperfect fault-handling $Q(t|\underline{\ell})$, or/and the probability of successful operation, $P(t|\underline{\ell})$.[9] A discussion of these output options with examples is given in section 4.0

1 - summary results only (default value)
2 - $P(t|\underline{\ell})$, probability of successful operation plus summary results
3 - $Q(t|\underline{\ell})$, probability of a fault-handling failure plus summary results
4 - all of the above

RLPLOT flag specifying if summary information QSUM, P*SUM and QSUM + P*SUM are to be plotted against time. See appendix A-2 for details.

IAXSRL specifies axes for summary information plot.

1 - Linear Y-axis
2 - Log Y-axis (default value)
3 - Linear and Log Y-axis
4 - Log-log plot only

---

8. For the example problem of section three, the CARE III default values for parameters LC, RLPLOT and IAXSRL will be used since these parameters are not set in the input file.

9. The user is cautioned in selecting options 2-4 for large systems. The number of $P(t|\underline{\ell})$ and $Q(t|\underline{\ell})$ values can be huge. By adjusting the control parameters, the user can exercise some control on the size of the output. See appendix F.

3.3 FAULT OCCURRENCE

NAMELIST syntax:

```
$FLTCAT     NFCATS(1)     = 1,
            NFCATS(2)     = 1,
            NFCATS(3)     = 1,
            JTYP(1,1)     = 1,
            RLM(1,1)      = 2.2E-4,
            JTYP(1,2)     = 1,
            RLM(1,2)      = 3.5E-5,
            JTYP(1,3)     = 2,
            RLM(1,3)      = 1.5E-6$
```

List-directed syntax:

```
1, 1, 1,
      1,      1,      1,
      ,       ,       ,
   2.2E-4, 3.5E-5, 1.5E-6/
```

Every fault type has a fault occurrence rate and fault-handling model associated with it. The fault-handling model associated with a fault type determines if the fault type is permanent, intermittent, or transient. Fault types are only identical if both their fault occurrence rate and fault-handling model are identical. The fault occurrence paragraph defines the quantity of fault types assigned to modules within a stage, the fault-handling model each fault type is linked to, and the fault type's corresponding fault occurrence rate. A stage may experience up to five different fault types.

$FLTCAT — fault occurrence input identifier which must appear at the start of the paragraph

NFCATS(x) — quantity of fault types assigned to stage x. The value of one for each stage implies that each stage has only one fault type. The fault type(s) may, however, differ from stage to stage, as it does in the example above.

JTYP(i,x) — links a fault-handling model to the ith fault type to stage x. JTYP(1,x) links the first fault type that affects stage x; JTYP(2,x) links the second fault type; etc. In the example above, stages 1 and 2 are both subject to the same fault-handling model, #1, and stage 3 is subject to fault-handling model, #2.

OMG(i,x) — parameter ω of the Weibull fault occurrence rate $\lambda\omega(\lambda t)^{\omega-1}$ for fault type i for stage x. The default value[10] for ω is 1.0,

---

10. The default values for the OMG parameters are used in the example problem of this section.

which yields a constant failure rate λ, an exponential distribution. The time scale is always in hours.

RLM(i,x) parameter λ of the Weibull fault occurrence rate for the fault type i for stage x. The time scale is always in hours.

3.4 RUN TIME

NAMELIST syntax:

```
$RNTIME     FT          = 10.0,
            SYSFLG      = .TRUE.,
            CPLFLG      = .TRUE.$
```

List-directed syntax:

```
10.0, , , .TRUE., .TRUE.,/
```

The RNTIME paragraph is used for specifying the operating time and the time scale for which the system is to be assessed. The user also specifies if a system fault tree is to follow which defines the system failure configurations and if a critical-pair fault tree is provided. Parameter controls are available which reduce the amount of computation.

$RNTIME run time input identifier which must appear at the start of the paragraph.

FT operating time for which the system is to be assessed.

NSTEPS minimum number of time steps[11] for which each $P(t|\underline{\ell})$ and $Q(t|\underline{\ell})$ probability array is listed for the operating time, FT, of the system. The number of time steps used affects the accuracy of the integration algorithms used; the more steps used, the greater the accuracy and computation time. If parameter LGTMST = .FALSE., then the number of time steps used will equal NSTEPS where the default is set to 50. If LGTMST = .TRUE., then the number of time steps used will be equal to or greater than NSTEPS and is determined automatically by CARE III.

---

11. For the example problem of this section, the CARE III default values for parameters NSTEPS, ITBASE, KWT, PSTRNC, CINDBG, QPTRNC, IVSN, NPSBRN, and CKDATA will be used since these parameters are not set in the input file.

ITBASE time scale used for system operating time (FT). This time scale is reflected only in the output listings and does not affect the scale of input parameters other than FT.

1 - hours
2 - minutes
3 - seconds
4 - milliseconds

SYSFLG flag stating if a system fault tree is to follow. The system fault tree states what stage failures define a system failure due to module depletion. The default system tree is a series fault tree; any stage failure causes system failure.

CPLFLG flag stating if a critical-pairs fault tree(s) is provided. The critical-pairs tree(s) defines which pairs of modules will cause a system failure due to coexisting faults (fault-handling unreliability, QSUM).

KWT this parameter is not currently used; however, its presence must be accounted for in the list-directed format.

PSTRNC parameter used to limit the number of fault vectors, $\underline{\ell}$, used in computing fault-handling unreliability. Fault vectors with a module depletion probability, $P^*(t|\underline{\ell})$ of less than PSTRNC will not enter into the fault-handling unreliability calculations. See appendix F for more detail.

CINDBG flag to create debug files in the CAREIN program module. This was used when developing the CARE III program.

QPTRNC parameter used to limit the number of fault vectors, $\underline{\ell}$, used in computing fault-handling unreliability, QSUM. The default value is 1.0E-2. Smaller values will cause more vectors to be assessed which have an impact on higher order terms in QSUM. See appendix F for more detail.

IVSN parameter selecting whether developmental version 3 or version 4 algorithms are to be used in the CARE3 program module.

NPSBRN parameter that is used to determined the number of subruns and consequently, the number of noncritically-paired stages that are to be grouped together in a subrun. The default is 20, but fastest execution will usually result if NPSBRN is three or four (see appendix D for a more detailed discussion).

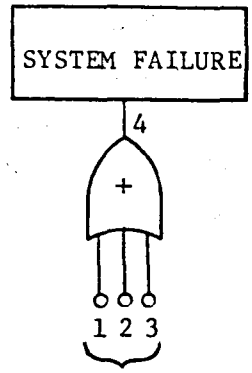
CKDATA flag indicating if the CARE III program is to check user's input before processing it. Default is CKDATA = .TRUE..

3.5 SYSTEM FAULT TREE

```
SYSTEM FAULT TREE FOR THREE STAGES IN SERIES     } TITLE
1 3 4 4                          } STAGE AND LOGIC RANGE ID
4 0 1 2 3                        } LOGIC GATE ID
```

The system failure description, in terms of stages, is given thru the system fault tree (see ref. 7 for fault tree nuances). All of the stage numbers from one up to the value of 'NSTGES' are used as input to this tree. Stage numbers take on physical significance. This means that stage number x used as an input to the system fault tree will use the input information directly associated with it, e.g., stage number 2 will use the input data M(2), N(2), NOP(1,2) thru NOP(5,2), NFCATS(2), JTYP(1,2) thru JTYP(5,2), RLM(1,2) thru RLM(5,2), and so on. The system fault tree is described using a small subset of logic operators depicted as logic gates. The system tree is constructed from these gates to define what combinations of the system stages, if failed, will produce system failure. The user is defining system failure states, H states as shown in figure 2-1, via the system fault tree. The system fault tree for the example discussed in section 3.0 is shown on the next page. The system fault tree has its own syntax and is independent of the syntax used for input paragraphs FLTTYP, STAGES, FLTCAT, and RNTIME. The rules for constructing a system tree are discussed subsequently.

Example



SYSTEM TREE

As shown at the top of this section (3.5), the system fault tree is divided into three parts; the TITLE section, the STAGE AND LOGIC RANGE ID section, and the LOGIC GATE ID section. Before creating these sections for the problem of interest, it is recommended that the user sketch his tree on paper to avoid numbering mistakes.

TITLE

identification label. The TITLE section is used to identify the system tree and to make any notes about it. This section of the system fault tree immediately follows the RNTIME input paragraph. To continue the TITLE section from one line to the next, the letter C preceded by a space must be typed as the last non-blank character between columns 9 and 80 inclusive. In this manner the TITLE section can be continued to as many lines as desired. The TITLE for the example problem of section 3.0 could have been written:

```
SYSTEM FAULT TREE          C
FOR THREE STAGES           C
IN SERIES
```

The system tree TITLE section is printed near the top of the output listing produced by program module CARE3 and therefore provides a useful mechanism for also identifying the computer listing and the system being modeled. See section four for example CARE III output.

STAGE AND LOGIC RANGE ID

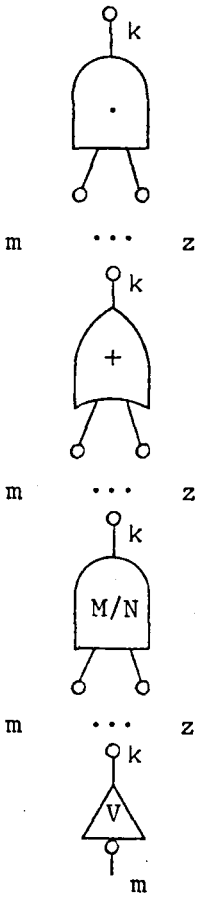
provides the range of inputs and range of logic gates defining a system fault tree. The information in this section is always contained in one

line. The line consists of four integers all separated by at least one space and is of the form 1 K I J. 1 and K identify the range of basic inputs to be used for the system fault tree. Each input corresponds to a physical stage of the system. The integers I and J identify the range of numbers associated with logic gates used to construct the system fault tree. A logic gate is numbered by assigning a unique integer to the logic gate's output. Thus the system tree will have to be constructed and numbered before this section can be completed. In all cases, however, the integers K, I and J must obey the relationship: NSTGES = K < I ≤ J ≤ 2000, where NSTGES is one of the input variables to input paragraph STAGES.

For the example, the first two integers of the STAGE AND LOGIC RANGE ID section must be 1 3 because NSTGES equals three. The system fault tree consists of one OR gate whose output is numbered 4. Since the system fault tree consists of a single gate, integers I and J are both 4. An example where I and J are not equal can be seen in appendix G.2. Because logic gates in a system fault tree must be numbered sequentially, the quantity J-I+1 equals the number of gates the tree contains.

LOGIC GATE ID

lists the logic gate descriptors for the system fault tree. There will be one descriptor for every gate of the tree. Logic gate descriptor format is of the form k l m n ... z, where k is the gate number, l is the gate type, and m n ... z are the gate input numbers which the gate operates on. Input events identify stages or outputs from preceding gates. In the example for this section, k = 4 (gate number), l = O (letter O for OR operator), and m n ... z = 1 2 3. Thus the logic gate descriptor is 4 O 1 2 3. The logic operators available are AND, OR, M of N, and INVERT. The general format for the logic gate descriptors is shown below. Notice that the OR gate can be represented by three different symbols. For the M of N gate, M is not the symbol used. M represents an integer larger than one and less than or equal to the number of inputs to the gate. For all gates used, k must be less than or equal to 2000, and inputs m n ... z must all be less than k.



AND	k A m n ... z	all events m thru z must occur (fail) for event k to occur
OR	k O m n ... z k O m n ... z k 1 m n ... z	if any of the events m thru z occur (fail), event k occurs
M of N	k M m n ... z	if 'M' of the events m thru z occur (fail), event k occurs
INVERT	k V m	if input m occurs, output k ceases to occur

3.5.1 System Fault Tree Rules and Procedures

In order for the CARE III program to read system fault tree information, the parameter SYSFLG in input paragraph RNTIME must be set equal to .TRUE.. If SYSFLG = .FALSE., then the CARE III program assumes that the user has not provided a system fault tree for input. In this case, CARE III assumes the default system tree. The default system tree is a single OR gate with all stage numbers serving as inputs. Note that for the discussion example this approach could have been taken.

The basic input numbers to a system fault tree are the stage numbers one thru 'NSTGES'. If any of these inputs are not used, the CARE III program assumes the omitted stages will never fail the system being modeled with respect to depletion of hardware. Another way to model this situation is to include such stages in the system fault tree and then setting the stage's M values in input paragraph STAGES to equal zero. In either case, the stages modeled may

contribute to system failure due to improper fault-handling, but the stages will not contribute to unreliability due to depletion of hardware. This feature was included to enable the user to model the effects of software errors. In this sense, software errors may lead to single or double fault system failures.

Logic gates of the system fault tree must be numbered sequentially without interruption. All numbers used must be integers. The inputs to any logic gate must be less than the gate number they feed. This requirement prevents the user from creating feedback paths in the tree. It is permitted, however, to let a basic input or gate output to act as an input to more than one higher level gate. This basic input event or gate output event is called a common mode event. As an example, the output to a gate numbered 105 could act as an input to any number of gates numbered greater than 105.

The first logic gate of a system fault tree does not have to have a value equal to 'NSTGES' + 1. A gap left between the highest numbered input (equal to NSTGES) and the lowest numbered logic gate is usually desired when numbering the fault tree. Leaving a gap of integers between the highest numbered input event and the lowest numbered logic gate will sometimes facilitate potential revisions without having to renumber the whole system fault tree.

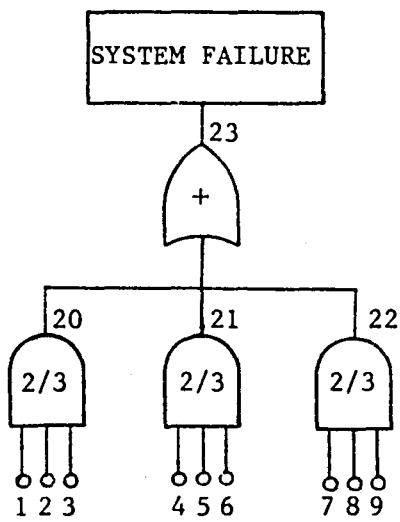
The addition of the INVERT gate for use in the system fault tree has been a recent addition to the CARE III program. The user is cautioned in its use. With the INVERT gate, it is possible to construct a system fault tree where no combination of failed stages will cause the system to fail (incoherent fault tree). The user is reminded that the system fault tree describes the events that cause system failure and not system success.

3.6 CRITICAL-PAIR FAULT TREE

```
CRITICAL-PAIR FAULT TREE      } TITLE
1 13 20 23                    } MODULE AND LOGIC RANGE ID
1 1 6                         }
2 7 11                        } MODULE UNIT TO STAGE ASSOCIATION
3 12 13                       }
20 2 1 2 3                    }
21 2 4 5 6                    } LOGIC GATE ID
22 2 7 8 9                    }
23 0 20 21 22
```

The inputs to a critical-pair tree represent functional modules from particular physical stages. The critical pair fault tree paragraph as illustrated above defines the pairs of functional modules from specific stages that have the potential to cause system failure. By including one or more critical-pair trees for the problem being analyzed, the CARE III program invokes the double-fault model illustrated in appendix C. The meaning of a critical-pair tree differs significantly from a system fault tree. The combinations of faulty pairs of modules are assumed to cause failure due to improper fault handling versus due to depletion of hardware. The construction of a critical-pair tree also differs significantly than that of the system fault tree. The critical-pair tree has an added (MODULE UNIT TO STAGE ASSOCIATION) section and there are more restrictions for the type of logic gates that can be used to build a critical-pair tree. The sections of the critical-pair tree and the rules for creating one or more critical-pair trees are discussed subsequently. The critical pair tree diagram for the example problem is shown below.

Example,



CRITICAL-PAIR TREE

TITLE

identification label for critical-pair tree. This label can be continued to as many lines as desired by using the same continuation rules as given for the system fault tree TITLE section.

MODULE AND LOGIC RANGE ID

provides the range of possible inputs and range of logic gates defining a critical-pair tree. The information in this section is always contained on one line. This line consists of four integers all seperated by at least one space and is of the form K L I J. Integers K and L represent the range of functional module numbers which may be used as inputs to the critical-pair tree (not all modules must be identified in the tree). Integers I and J identify the range of logic gate numbers to be used in the critical-pair tree. The four integers of this section must obey the following relationships; $(L-K+1) \leq 70$ and $1 \leq K \leq L < I \leq J \leq 2000$. The first relation restricts the number of basic input numbers to 70 and the second relation restricts the numbering of logic gates to be less than or equal to 2000. For the example of this section, there is a total of (13-1+1) or 13 modules whose stages will be formed into a subrun (see below) by the critical-pair tree, and the logic statements for the tree start with number 20 and end with number 23.

MODULE UNIT TO STAGE ASSOCIATION

shows the correspondence of functional module input numbers to a physical stage. The form of a particular line of this section is X B C. X is the number of the physical stage that consists of functional modules numbered B thru C. The stage numbers X from one line to the next do not have to be consecutive but must be in increasing order. The range of stage numbers used by this section cannot exceed twenty. Thus the MODULE UNIT TO STAGE ASSOCIATION section will not be longer than twenty lines.

In the example problem, modules 1-6 are associated with processor stage 1, modules 7-11 with memory stage 2, and modules 12-13 with power supply stage 3. All three MODULE UNIT TO STAGE ASSOCIATION statements are listed in the critical-pair tree paragraph even though the critical-pair tree diagram does not show modules 12 and 13. The statement, 3 12 13, is included so that only one subrun will be formed; otherwise,

if it were omitted and it could have been, then two subruns would have been formed to produce the same results (see section 3.6.1 and appendix D.2). Creating two subruns, however, would introduce additional difficulty in describing the CARE III output, discussed later. Modules 10 and 11 although listed are not critically coupled. They are listed in statement 2 7 11 to simply associate them with stage 2.

The integer B of the first MODULE UNIT TO STAGE ASSOCIATION line must equal the integer K of the MODULE AND LOGIC RANGE ID section. For the rest of the MODULE UNIT TO STAGE ASSOCIATION lines, the integer B must equal one more than the integer C of the preceding line. The integer C of the last MODULE UNIT TO STAGE ASSOCIATION line must equal the integer L of the MODULE AND LOGIC RANGE ID section. Thus all L-K+1 functional module numbers are associated with some stage.

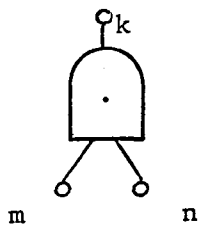
At the beginning of a mission, each stage X of the system under study consists of N(X) modules. (See section 3.2 for N(x) description.) Another rule in constructing the MODULE UNIT TO STAGE ASSOCIATION section is that the quantity C-B+1 for each line must equal N(X). For the example of this section, notice that 6-1+1 = N(1), 11-7+1 = N(2), and 13-12+1 = N(3).

When more than one critical-pair tree is created by the user, the stage numbers of the MODULE UNIT TO STAGE ASSOCIATION section must be in increasing order from one tree to the next. Thus the same stage cannot be referenced by more than one critical-pair tree. The input numbers to a critical-pair tree can, however, be used again from one tree to the next since they have functional, versus physical significance.

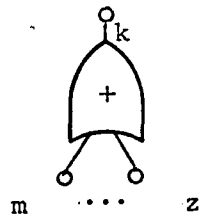
LOGIC GATE ID

This section lists the logic gate descriptors for the critical-pair tree. There will be one descriptor for every gate of the tree. Logic gate descriptor format is identical to that of the system fault tree LOGIC GATE ID section. Fewer gates are available for making a critical-pair tree than for a system fault tree. The gates available are 2-input AND gates, 2 of N gates, and OR gates with two or more inputs. Input events to these gates are functional module numbers or outputs from preceding

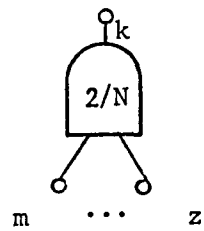
gates. The general format for these logic gates are shown below. Notice that the OR gate can be represented by three different symbols. For each gate used, k must be less than or equal to 2000, and inputs m n ... z must all be less than k.



AND	k A m n	if events m and n occur (fail), event k occurs



OR	k O m n ... z k O m n ... z k 1 m n ... z	if any of the events m thru z occur (fail), event k occurs



2 of N	k 2 m n ... z	if any two of the events m thru z occur (fail), event k occurs

3.6.1 Critical-Pair Tree Rules and Procedures

Critical-pair tree information is read and processed only if the parameter CPLFLG in input paragraph RNTIME is set equal to .TRUE.. If CPLFLG = .FALSE., and if all fault-handling models used for the system being modeled all have their C parameters in input paragraph FLTTYP set to one, then all $Q(t|\underline{\ell})$ and QSUM values calculated and shown on the CARE III output listing will equal zero.

Logic gates for a critical-pair tree must be numbered sequentially without interruption. All numbers used must be integers. The inputs to any logic gate must be less than the gate they feed. The first input module number does not have to equal one, nor does the first logic gate have to be numbered one more than the last input module number. Like the system fault tree, leaving a gap

of integers between the highest numbered input event and the lowest numbered logic gate will sometimes facilitate revisions without having to renumber the whole critical-pair tree.

The purpose of a critical-pair tree is to indicate the pairs of module failures (input events) that can cause system failure due to improper fault handling. A critical-pair tree should not be constructed so that one, or more than two module failures cause an output from the top most gate. To facilitate this type of construction, the CARE III program only allows 2-input AND gates, 2 of N gates, and OR gates with two or more inputs when constructing a critical-pair tree. Combinations of three or more events which lead to the top event of a critical-pair tree are ignored by CARE III. For the example of this section, any two faults among modules (functionally numbered) 1, 2, and 3, or any two among modules 4, 5, and 6 cause failure of the first or second triad as well as the system. From the example, the reader should note that not all modules need to be critically coupled, only those that apply to the system design. More examples of critical-pair trees can be found in appendix G.

CARE III often computes fault-handling unreliability in segments called subruns. Every critical-pair tree created by the user will spawn a subrun. For details on how this works see appendix D.2. The example problem of this section is suitable for partitioning into subruns. Below are two examples of critical-pair tree input that would create two subruns from the discussion problem. Stage one would be in subrun one, and stages two and three would be in subrun two. Note that many other types of tree numbering are possible than the ones shown.

Example one:

```
CRITICAL-PAIR TREE ONE
 1  6  7  9
 1  1  6
 7  2  1  2  3
 8  2  4  5  6
 9  0  7  8
CRITICAL-PAIR TREE TWO
 1  7  8  8
 2  1  5
 3  6  7
 8  2  1  2  3
```

Example two:

```
CRITICAL-PAIR TREE ONE
11 16 21 23
1 11 16
21 2 11 12 13
22 2 14 15 16
23 1 21 22
CRITICAL-PAIR TREE TWO
21 27 31 31
 2 21 25
 3 26 27
31  2 21 22 23
```

3.7 INPUT SUMMARY

The table on the next page provides the user with a quick-look summary of all the user input values and their valid ranges. The variables listed in the first column are the user definable input variable names that are presented as arrays. The array size is given by column two and tells the user the maximum number of values that can be defined for its corresponding variable name (Col. 1). The variable and arrays are defined in sections 3.1-3.4. The variables and arrays are listed in the table in the order in which they must be provided for list-directed type formatting.

Because of the possible large amount of input data that can be described, CARE III uses default parameters. The default values will be automatically used by CARE III unless another value is defined. The default values have been preselected by the CARE III designers as their best guess of the values that would most likely be used most often. The right most column defines the valid ranges of CARE III input values, which means, CARE III will not give a warning or abort the execution if user specified values are within range. The listed values are those that CARE III will accept as possible input; however, the reasonableness of using sets of these values must be obtained from engineering judgement. With this approach, correct execution is virtually assured. A random selection of values from this list will not guarantee correct execution. Typically, the fault-handling rate parameters should be three or four orders of magnitude larger than the fault occurrence rates. A 32 bit machine will not be able to process as large a range of fault-handling parameters as a 60 or 64 bit machine will. If fault-handling parameters are too large for a given CARE III problem, the COVRGE program module will write an error message to the output listing.

VARIABLE	ARRAY SIZE	DEFAULT	RANGE
FLTTYP - fault-handling parameters			
NFTYPS	1	1	1 - 5
ALP(i)	5	0.0	≥ 0.0
BET(i)	5	0.0	≥ 0.0
DEL(i)	5	3.6E+3	≥ 0.0
RHO(i)	5	0.0	≥ 0.0
EPS(i)	5	0.0	≥ 0.0
IDELF(i)	5	1	1 or 2
IRHOF(i)	5	1	1 or 2
IEPSF(i)	5	1	1 or 2
PA(i)	5	1.0	0.0 - 1.0
PB(i)	5	0.0	0.0 - 1.0
C(i)	5	1.0	0.0 - 1.0
DBLDF	1	.05	0.01 - 0.1
TRUNC	1	1.0E-4	1.0E-3 - 1.0E-6
CVPRNT	1	.FALSE.	.FALSE. or .TRUE.
CVPLOT	1	.FALSE.	.FALSE. or .TRUE.
IAXSCV	1	2	1 - 4
MARKOV	1	1	1 or 2
LGTMST	1	.TRUE.	.FALSE. or .TRUE.
STAGES - stage definition parameters			
NSTGES	1	1	1 - 70
N(x)	70	1	≥ 1
M(x)	70	1	≥ 0
NOP(j,x)	(5,70)	see NOP on page 27	
LC(x)	70	0	0 - N(x)
IRLPCD	1	1	1 - 4
RLPLOT	1	.FALSE.	.FALSE. or .TRUE.
IAXSRL	1	2	1 - 4
FLTCAT - fault occurrence parameters			
NFCATS(x)	70	1	1 - 5
JTYP(j,x)	(5,70)	1	1 - 5
OMG(j,x)	(5,70)	1.0	> 0.0
RLM(j,x)	(5,70)	1.0E-4	> 0.0
RNTIME - run time parameters			
FT	1	1.0	> 0.0
NSTEPS	1	50	17 - 64
ITBASE	1	1	1 - 4
SYSFLG	1	.FALSE.	.TRUE. or .FALSE.
CPLFLG	1	.FALSE.	.TRUE. or .FALSE.
KWT	1	(not used)	(not used)
PSTRNC	1	1.0E-14	> 0.0
CINDBG	1	.FALSE.	.TRUE. or .FALSE.
QPTRNC	1	1.0E-2	> 0.0
IVSN	1	4	3 or 4
NPSBRN	1	20	1 - 20
CKDATA	1	.TRUE.	.TRUE. or .FALSE.

3.8 AVAILABILITY OF DATA

The acquisition of reliability failure data is a perennial problem for analysts even when traditional piece-part reliability models are used. The most common source of data is given by Mil-Handbook 217. Unfortunately not much data are available for Weibull failure distributions. Over the years, several papers have appeared in the literature supporting the use of Weibull distributions, e.g., see references 8, 9, 10, and 11; however, the use of Weibull distributions for prediction of system reliability remains sparse.

The justification for including it in CARE III is manifold: As a consequence of the state aggregation/decomposition technique that enables CARE III to handle very large state spaces, CARE III implements a nonhomogenous Markov model which uses nonconstant transition rates. The Weibull distribution is a very flexible distribution with a nonconstant hazard rate that reduces to an exponential distribution (constant failure rate) by setting one parameter. The Weibull distribution permits the analyst the opportunity to examine the effects of wear-out on the reliability estimate, or it can be used to model modules that are redundant internally. Note that CARE III normally models replication of simplex modules in a stage, and redundancy across stages can also be modeled. Since the distributions of several CARE III model parameter distributions are not well known (e.g., transient, intermittent, and software faults), it is fitting to have a generalized distribution available (i.e., Weibull distribution).

Some work was done to estimate intermittent arrival time distributions (ref. 12) and to characterize transient faults (ref. 13). New work on predicting software reliability is described in reference 14; and references 15 and 16 present fault latency data for permanent faults.

4.0 OUTPUT

Each of the CARE III program modules writes data to an output file listing. This listing and its interpretation is discussed in this section.

4.1 CAREIN Output

CAREIN is the first main program module of the CARE III computer program. The CAREIN module preprocesses the user-supplied input file for program modules COVRGE and CARE3. The CAREIN module begins by copying the user-supplied input file to the output listing. The CARE III header is printed next. The header is followed by a list of the stages for the system under study. For each stage of the system, the probability of failure due to permanent and intermittent faults[12] is calculated at time, t, where t is the end mission time specified by input parameters 'FT' and 'ITBASE'. This list of probabilities is provided to give the user a 'feel' for the minimum probability of failure for each stage, when each stage is considered separately. These probabilities can be used to determine if a reasonable P*SUM computation is presented, when the following conditions exist: For long mission times, it is possible that the CARE III program will not compute P*SUM conservatively for some systems containing twelve or more stages. This condition may occur when the probability of module depletion failure for one or more stages approaches one, leading to a P*SUM value that approaches one. Normally, this condition will not occur when modeling ultrareliable systems. If it should occur, a warning message is printed by the CAREIN program module to alert the user.

A second list is provided by the CAREIN program module. This list shows how many subruns will be created by the CARE3 program module, the physical stages that are considered in each subrun, and the subruns if any, that will involve critical-pair failure calculations (see appendix D.2 for information on how subruns are created and what they are).

If the CAREIN module executes successfully, a "SUCCESSFUL EXECUTION OF CAREIN" message will be printed. If this message does not appear, program modules COVRGE and CARE3 will not execute normally.

---

12. Transient faults are not accounted for in this list.

4.2 COVRGE Output

The COVRGE program module prints a CARE III header followed by a table listing the parameters for all fault-handling models used by the current problem. Information for at least one fault-handling model will be shown. If input parameter CVPRNT is set .TRUE., some debugging information will also be printed on the listing. The debug information, if printed, is usually of little or of no interest to a typical user.

4.3 CARE3 Output

It is the CARE3 program module output which provides the unreliability predictions for the system being modeled. Predictions for overall system unreliability along with general information about subruns is always given, but specific information about individual subruns may also be obtained. This subrun information will be described subsequently and will be followed with a discussion of the summary information.

Like the COVRGE program module, the CARE3 program module begins by printing a CARE III header to the output listing. The header is followed by the system fault tree title, provided a system fault tree existed in the user-supplied input file to the CAREIN program.

The next portion(s) of CARE3 output provides subrun information. Every CARE III problem will contain at least one subrun. Part of the listing for the example of section three is shown as figures 4-1 thru 4-3 showing the structure of the subrun portion of the listing. The beginning of a subrun listing starts with a subrun number. This number is printed at the top of a fresh page. Following the subrun number is the range of stages included in the subrun. The number of stages in this range determines how many elements the subrun's $\underline{\ell}$ vector will have. Next in the listing is a table summarizing each fault occurrence rate and fault-handling model used by the stages in the subrun. When looking at this table, note that 'ICAT' is used for bookkeeping purposes to identify the ith fault type that a stage is subject to, and 'JTYP' represents a fault-handling model number.

SUBRUN 1

STAGES 1 - 3

CONFIGURATION:

STAGE	N	M
1	6	2
2	5	2
3	2	1

FAULT INFORMATION:

ICAT	RLM	OMG	JTYP
1	2.200E-04	1.00E+00	1
1	3.500E-05	1.00E+00	1
1	1.500E-06	1.00E+00	2

FAULT VECTORS:

TIMES AT WHICH THE FOLLOWING FUNCTION VALUES CORRESPOND (IN HOURS):

0.0000000000E+00	3.8147554733E-05	7.6295109466E-05	1.1444266420E-04	1.5259021893E-04
2.2888532840E-04	3.0518043786E-04	3.8147554733E-04	4.5777065679E-04	6.1036087573E-04
7.6295109466E-04	9.1554131359E-04	1.0681315325E-03	1.3733119704E-03	1.6784924082E-03
1.9836728461E-03	2.2888532840E-03	2.8992141597E-03	3.5095750354E-03	4.1199359111E-03
4.7302967869E-03	5.9510185383E-03	7.1717402898E-03	8.3924625069E-03	9.6131842583E-03
1.2054627761E-02	1.4496071264E-02	1.6937514767E-02	1.9378958270E-02	2.4261845276E-02
2.9144732282E-02	3.4027621150E-02	3.8910508156E-02	4.8676282167E-02	5.8442056179E-02
6.8207830191E-02	7.7973604202E-02	9.7505152225E-02	1.1703670025E-01	1.3656824827E-01
1.5609979630E-01	1.9516289234E-01	2.3422598839E-01	2.7328908443E-01	3.1235218048E-01
3.9047837257E-01	4.6860456467E-01	5.4673075676E-01	6.2485694885E-01	7.8110933304E-01
9.3736171722E-01	1.0936141014E+00	1.2498664856E+00	1.5623712540E+00	1.8748760223E+00
2.1873807907E+00	2.4998855591E+00	3.1248950958E+00	3.7499046326E+00	4.3749141693E+00
4.9999237061E+00	6.2499427795E+00	7.4999618530E+00	8.7499809265E+00	1.0000000000E+01

P(0, 0, 0)

1.0000000000E+00	1.0000000000E+00	1.0000000000E+00	1.0000000000E+00	9.9999964237E-01
9.9999964237E-01	9.9999964237E-01	9.9999964237E-01	9.9999928474E-01	9.9999928474E-01
9.9999892712E-01	9.9999862909E-01	9.9999827147E-01	9.9999791384E-01	9.9999755621E-01
9.9999719858E-01	9.9999684095E-01	9.9999547005E-01	9.9999475479E-01	9.9999403954E-01
9.9999302626E-01	9.9999123812E-01	9.9998950958E-01	9.9998742342E-01	9.9998569489E-01
9.9998217821E-01	9.9997800589E-01	9.9997448921E-01	9.9997097254E-01	9.9996352196E-01
9.9995619059E-01	9.9994885921E-01	9.9994152784E-01	9.9992686510E-01	9.9991250038E-01
9.9989771843E-01	9.9988305569E-01	9.9985402822E-01	9.9982458353E-01	9.9979555607E-01
9.9976611137E-01	9.9970763922E-01	9.9964880943E-01	9.9959063530E-01	9.9953222275E-01
9.9941533804E-01	9.9929809570E-01	9.9918121099E-01	9.9906438589E-01	9.9883025885E-01
9.9859708548E-01	9.9836137725E-01	9.9812924862E-01	9.9766230583E-01	9.9719524384E-01
9.9672865868E-01	9.9626195431E-01	9.9532979727E-01	9.9439829588E-01	9.9346780777E-01
9.9253785610E-01	9.9068117142E-01	9.8882830143E-01	9.8697817326E-01	9.8513162136E-01

Q(0, 0, 0)

Figure 4-1 Operational and Fault-Handling Failure Probabilities, Example from Section 2.0

Figure 4-2. Operational and Fault-Handling Failure Probabilities (con't), Example from Section 2.0

```
          Ø.Ø FOR ALL TIME STEPS.

P( Ø, Ø, 1)
          Ø.ØØØØØØØØØØØE+ØØ 1.1444258735E-1Ø 2.2888528572E-1Ø 3.4332825471E-1Ø 4.5777Ø62696E-1Ø
          6.8665534370E-1Ø 9.1554169801E-1Ø 1.1444259984E-Ø9 1.3733107984E-Ø9 1.8310800654E-Ø9
          2.2888515527E-Ø9 2.7466211527E-Ø9 3.2043878662E-Ø9 4.1199248457E-Ø9 5.Ø3546893Ø7E-Ø9
          5.9510014694E-Ø9 6.8665371167E-Ø9 8.6976053Ø26E-Ø9 1.Ø528673Ø45E-Ø8 1.2359744339E-Ø8
          1.4190788100E-Ø8 1.7852890721E-Ø8 2.1514994231E-Ø8 2.5177Ø92411E-Ø8 2.8839158617E-Ø8
          3.6163243067E-Ø8 4.3487268897E-Ø8 5.Ø811244989E-Ø8 5.8135185554E-Ø8 7.2782952998E-Ø8
          8.7430407802E-Ø8 1.Ø2Ø7762102E-Ø7 1.1672473477E-Ø7 1.4601806697E-Ø7 1.7531074548E-Ø7
          2.Ø460252870E-Ø7 2.3389343Ø85E-Ø7 2.9247291877E-Ø7 3.5104855556E-Ø7 4.Ø9621Ø3753E-Ø7
          4.6818982469E-Ø7 5.8531753666E-Ø7 7.Ø243146411E-Ø7 8.1953180597E-Ø7 9.3661839173E-Ø7
          1.1707512613E-Ø6 1.4048277990E-Ø6 1.6388496533E-Ø6 1.8728184159E-Ø6 2.3405887077E-Ø6
          2.8081421988E-Ø6 3.2754744552E-Ø6 3.7425879782E-Ø6 4.6761633712E-Ø6 5.6Ø886155Ø5E-Ø6
          6.54Ø6879912E-Ø6 7.4716390372E-Ø6 9.3309217846E-Ø6 1.1186728443E-Ø5 1.3039Ø5537Ø5E-Ø5
          1.4887895304E-Ø5 1.8575197828E-Ø5 2.2248652385E-Ø5 2.59Ø83ØØ813E-Ø5 2.9554181310E-Ø5

Q( Ø, Ø, 1)
          Ø.Ø FOR ALL TIME STEPS.

P( Ø, 1, Ø)
          Ø.ØØØØØØØØØØØE+ØØ 6.6758198969E-Ø9 1.3351646899E-Ø8 2.ØØ27451697E-Ø8 2.67Ø3293798E-Ø8
          4.ØØ549Ø3394E-Ø8 5.34Ø6512990E-Ø8 6.6758211403E-Ø8 8.Ø1Ø982Ø999E-Ø8 1.Ø6813Ø4730E-Ø7
          1.3351639438E-Ø7 1.6Ø219642ØØE-Ø7 1.8692270487E-Ø7 2.4Ø329Ø5799E-Ø7 2.9373526900E-Ø7
          3.4714193475E-Ø7 4.ØØ54817418E-Ø7 5.Ø736Ø48252E-Ø7 6.1417222241E-Ø7 7.2Ø98464443E-Ø7
          8.2779638433E-Ø7 1.Ø414190683E-Ø6 1.2550419797E-Ø6 1.4686629584E-Ø6 1.6822833686E-Ø6
          2.1Ø952362Ø7E-Ø6 2.5367567105E-Ø6 2.9639913919E-Ø6 3.3912215258E-Ø6 4.2456713345E-Ø6
          5.1ØØ1Ø61365E-Ø6 5.9545332Ø78E-Ø6 6.8Ø89429988E-Ø6 8.5177307483E-Ø6 1.Ø226474842E-Ø5
          1.1935165276E-Ø5 1.3643Ø86597E-Ø5 1.7Ø6Ø938262E-Ø5 2.Ø477880753E-Ø5 2.3894624974E-Ø5
          2.7311154554E-Ø5 3.4143646189E-Ø5 4.Ø975304728E-Ø5 4.7806250222E-Ø5 5.4636377172E-Ø5
          6.8294219Ø91E-Ø5 8.1948877778E-Ø5 9.56ØØ466Ø11E-Ø5 1.Ø924882372E-Ø4 1.3653606584E-Ø4
          1.6381Ø9Ø973E-Ø4 1.9107288972E-Ø4 2.1832228231E-Ø4 2.7278313064E-Ø4 3.2719364390E-Ø4
          3.8155386574E-Ø4 4.3586362153E-Ø4 5.4433243349E-Ø4 6.526ØØ31261E-Ø4 7.6Ø666793Ø1E-Ø4
          8.6853356333E-Ø4 1.Ø836659931E-Ø3 1.298ØØ13853E-Ø3 1.5115380520E-Ø3 1.7242822796E-Ø3

Q( Ø, 1, Ø)
          Ø.Ø FOR ALL TIME STEPS.

P( 1, Ø, Ø)
          Ø.ØØØØØØØØØØØE+ØØ 5.Ø3547710Ø0E-Ø8 1.ØØ7Ø959178E-Ø7 1.5106425622E-Ø7 2.Ø1419211Ø8E-Ø7
          3.Ø212854Ø85E-Ø7 4.Ø283785552E-Ø7 5.Ø3547823Ø8E-Ø7 6.Ø4257138Ø5E-Ø7 8.Ø5675711Ø4E-Ø7
          1.ØØ7Ø945109E-Ø6 1.2Ø85130265E-Ø6 1.4Ø99317696E-Ø6 1.8127682324E-Ø6 2.2156Ø59458E-Ø6
          2.6184407Ø34E-Ø6 3.Ø212790989E-Ø6 3.8269463403E-Ø6 4.6326163101E-Ø6 5.4382885537E-Ø6
          6.2439539761E-Ø6 7.8552811829E-Ø6 9.4666Ø20232E-Ø6 1.1Ø779274Ø7E-Ø5 1.2689238247E-Ø5
          1.5911844457E-Ø5 1.9134422473E-Ø5 2.2356982299E-Ø5 2.5579529392E-Ø5 3.2Ø24541724E-Ø5
          3.8469490391E-Ø5 4.4914344471E-Ø5 5.1359Ø85774E-Ø5 6.4248335548E-Ø5 7.7137279732E-Ø5
          9.ØØ258382Ø0E-Ø5 1.Ø2914Ø5488E-Ø4 1.2868944Ø32E-Ø4 1.5446329780E-Ø4 1.8Ø2359Ø383E-Ø4
          2.Ø6ØØ715652E-Ø4 2.5754538365E-Ø4 3.Ø9Ø7767359E-Ø4 3.6Ø6Ø463754E-Ø4 4.1212642100E-Ø4
          5.1515229279E-Ø4 6.1815575464E-Ø4 7.2113738861E-Ø4 8.24Ø96Ø8876E-Ø4 1.Ø2994672Ø9E-Ø3
          1.2357Ø87107E-Ø3 1.4413804747E-Ø3 1.6469646944E-Ø3 2.Ø578624681E-Ø3 2.4684Ø36616E-Ø3
          2.8785897885E-Ø3 3.2884182874E-Ø3 4.1Ø7ØØ96195E-Ø3 4.9241785891E-Ø3 5.7399254292E-Ø3
          6.5544289537E-Ø3 8.1788Ø5ØØ11E-Ø3 9.7972750664E-Ø3 1.141Ø55Ø214E-Ø2 1.3Ø17944992E-Ø2

Q( 1, Ø, Ø)
          Ø.Ø FOR ALL TIME STEPS.

P( Ø, 1, 1)
          Ø.ØØØØØØØØØØØE+ØØ 7.6399812752E-19 3.Ø559956120E-18 6.8759Ø1270E-18 1.2223988238E-17
          2.75Ø3922458E-17 4.8895Ø06631E-17 7.6399863831E-17 1.1ØØ1576262E-16 1.9558338830E-16
          3.Ø559953473E-16 4.4ØØ6326225E-16 5.9897389588E-16 9.9Ø139693Ø9E-16 1.479Ø984616E-15
```

THE FOLLOWING $Q(t|\underline{\ell})$ VALUES ARE NOT SHOWN IN DETAIL:

$Q(1,0,1) = 0$	$Q(1,1,0) = 0$	$Q(2,0,0) \geq 0$	$Q(0,2,1) \geq 0$
$Q(0,3,0) \geq 0$	$Q(1,1,1,) = 0$	$Q(1,2,0) \geq 0$	$Q(2,0,1) \geq 0$
$Q(2,1,0) \geq 0$	$Q(3,0,0) \geq 0$	$Q(0,3,1) \geq 0$	$Q(1,2,1) \geq 0$
$Q(1,3,0) \geq 0$	$Q(2,1,1) \geq 0$	$Q(2,2,0) \geq 0$	$Q(3,0,1) \geq 0$
$Q(3,1,0) \geq 0$	$Q(4,0,0) \geq 0$		

$Q(X,Y,Z) = 0$ means, all values are zero for time $t = 0$ to specified mission time.

$Q(X,Y,Z) \geq 0$ means, only the $t = 0$ value is zero, all others are greater than zero.

Figure 4-3. Fault-handling failure probabilities, example from section 2.0.

The division of stages into subruns is a function of two factors, the presence of one or more critical-pair trees and the parameter NPSBRN. If a critical-pair tree was included, it will determine the subrun structure. If a critical-pair tree is not included, then the parameter NPSBRN determines the petitioning of stages in a subrun (see appendix D). If the latter case applies and the user is only interested in seeing P*SUM values in the summary; even though no $P(t|\underline{\ell})$ arrays are requested, they will nevertheless be computed. Therefore, it is essential to use NPSBRN wisely. This point is illustrated in the following discussion.

Figure 2-1 shows the general structure of the CARE III aggregate model. The vector $\underline{\ell}$ for this figure can have a maximum of 70 components, and, therefore, this figure depicts a model of up to 70 dimensions (see section 2.1). To simplify the discussion, let $\underline{\ell}$ have two components, $\ell(1)=i$ (stage one has i faults) and $\ell(2)=j$ (stage two has j faults) so that $\underline{\ell}=(i,j)$. The aggregate model for $\underline{\ell}=(i,j)$ is, therefore, two dimensional. In the context of CARE III subruns, two possibilities exist: one subrun can be formed where $\underline{\ell}=(i,j)$, or two subruns can be formed so that for subrun one, $\underline{\ell}=(i)$, and for subrun two, $\underline{\ell}=(j)$. From a computational point of view, the consequences of specifying one or more subruns can in general be dramatic. Since i and j represent the number of failed modules per respective stage, where $0\leq i\leq K\leq 70$ and $0\leq j\leq L\leq 70$, the number of possible aggregate $G(\underline{\ell})$ states for $\underline{\ell}=(i,j)$ is $(K+1)\cdot(L+1)$. By dividing up the state space of $\underline{\ell}$ vectors into two subruns so that for subrun one $\underline{\ell}=(i)$ (the number of $G(\underline{\ell})$ states is K+1) and for subrun two $\underline{\ell}=(j)$ (the number of $G(\underline{\ell})$ states is L+1), a total of K+L+2 possible aggregate $G(\underline{\ell})$ states results.

Aside from illustrating the importance of using NPSBRN judicially, the discussion also gives an interpretation of $P(t|\underline{\ell})$ values in a subrun. Recall from section 2.1 that $P(t|\underline{\ell})$ is the probability of the system being in state $G(\underline{\ell})$ at time t. If two subruns were requested, then $P(t|\underline{\ell})$ for $\underline{\ell}=(i)$, $\underline{\ell}=(j)$ is the probability of the system being in state $G(\underline{\ell}=(i))$, $G(\underline{\ell}=(j))$, where the system has experienced exactly i faults in stage one, exactly j faults in stage two. Similarly if one subrun was requested, then $P(t|\underline{\ell})$ for $\underline{\ell}=(i,j)$ is the probability of the system being in state $G(\underline{\ell}=(i,j))$ where the system has experienced exactly i faults in stage one and j faults in stage two. The user ability to influence the formation of $P(t|\underline{\ell})$ probabilities into different subruns may be useful to those analysts interested in predicting the number of replacement modules required to effect repairs at various mission times.

The time points at which CARE III calculates its operational probability functions, $P(t|\underline{\ell})$, and its unreliability functions, $Q(t|\underline{\ell})$, QSUM, and P*SUM, are determined by several factors. This necessitates that the time points for which these calculations are solved to be shown on the output listing. Figure 4-1 shows the location and format of these time points. If a problem contains more than one subrun, the time points used from one subrun to the next will be the same. In all cases the first time point will be for t=0 and the final time point will be for the ending mission time specified by input parameters FT and ITBASE. In the event the user does not define a critical-pair tree, a warning message will appear following the time points listing: **WARNING- CRITICAL FAULT PAIR FILE 'BXYFL' .....**. The message is provided to alert the user that no critical-pair tree was read by CARE-III.

If input parameter IRLPCD is set equal to one, than the listing for each subrun in the user's problem will finish with the time array just described. If IRLPCD is set equal to four, then the listing for each subrun will also include arrays, $Q(t|\underline{\ell})$, the probabilities of fault-handling failure for fault vector $\underline{\ell}$, and $P(t|\underline{\ell})$ arrays, the probabilities of successful operation for fault vector $\underline{\ell}$. If IRLPCD equals three, then only $Q(t|\underline{\ell})$ arrays are printed; if IRLPCD equals two, then only $P(t|\underline{\ell})$ arrays are printed. For the example problem of section three, IRLPCD equals four. Some of the $P(t|\underline{\ell})$ and $Q(t|\underline{\ell})$ arrays for this problem are shown in figure 4-2.

In many CARE III applications, the user will be modeling a fault-tolerant system where the system can experience a number of faults before system failure. The $P(t|\underline{\ell})$ probabilities tell the analyst the probability the system or subsystem modeled by a subrun is still operational even though it has experienced $\ell(1),\ell(2)...$ faults. For the example shown, the subrun contains all stages of the system being modeled. The P(0,0,0) value at time step number fifty for the example means that at a mission time of 7.8E-1 hours[13] (46.9 minutes) the system will be operational with probability 9.988E-1, where no permanent or intermittent faults occurred in the processor stage, or memory stage, or power supply stage. $P(t|\underline{\ell})$ information may be of interest if one wants to predict the number of replacement modules required for maintenance at the end of a mission.

---

13. CARE III lists more than three significant digits, but due to the user's host machine, the user may only see agreement to about three significant digits when comparing his output listing to the one in this guide.

$Q(t|\underline{\ell})$ probabilities tell the analyst the probability of system failure due to $\ell(1),\ell(2),..$ faults having occurred in the subsystem or system modeled by the subrun in which the $Q(t|\underline{\ell})$ probabilities are listed. From figures 4-2 and 4-3, one notes that there are no single faults which cause a fault-handling failure for the example problem. This results because all of the $Q(t|\underline{\ell})$ values representing the occurrence of a singe stage fault, e.g., Q(0,0,1), Q(0,1,0), and Q(1,0,0), are zero for all time points. Only when more than one fault occurs do we see nonzero $Q(t|\underline{\ell})$ array values, e.g., Q(0,2,0) is greater than zero. Thus two faults occurring in close succession in stage two may defeat the system's fault handling strategy and cause the system to fail. This failure mechanism is indicated thru the critical-pair tree. For the example, the Q(0,0,2) array does not appear because two faults in stage three cause stage failure by module depletion as specified by the system fault tree. This failure mechanism will be accounted for in the P*SUM function yet to be discussed. Similarly Q(0,4,0) doesn't appear since the vector, (0,4,0), also represents failure due to depletion of modules and will be accounted for in the P*SUM function.

When interpreting $P(t|\underline{\ell})$ and $Q(t|\underline{\ell})$ array values, it must be remembered that transient faults are counted differently than permanent and intermittent faults. If the example problem above had considered transient faults, then the Q(0,0,0) array may not have been equal to zero for all time points.

When a CARE III problem is partitioned into more than one subrun, the additional subruns follow the same format as the first. For large problems, selecting an IRLPCD value of four may produce large amounts of output. In many cases, selecting IRLPCD equal to three will provide enough detail about potential weak points of a design while controlling the volume of output.

4.4 Summary Output from CARE3

The last portion of the CARE3 program module's output is the CARE III summary information. The summary gives information on functions QSUM, P*SUM, and QSUM + P*SUM, defined thus:

QSUM — probability of a system failure due to imperfect fault handling (fault-handling unreliability). QSUM is the sum of the $Q(t|\underline{\ell})$ functions over all subruns.

$$QSUM(t) = \sum_{s} \left[\sum_{\underline{\ell}_s} Q(t|\underline{\ell}_s) \right] \qquad \text{where } s$$

is an index indicating subruns, and $\underline{\ell}_s$ are fault vectors for subrun s.

P*SUM — probability of a system failure due to depletion of modules (module depletion unreliability).

QSUM + P*SUM — probability equal to the sum of the above two probabilities. The value computed is called the total system unreliability.

For the summary output of the example shown as figure 4-4, one notes that the fault-handling unreliability QSUM at 10.0 hours is 8.20E-10, and the module depletion unreliability P*SUM, is 2.25E-10.

A breakdown of the above two unreliability functions per exactly K stages having failed by the end of the mission time is given following the QSUM + P*SUM array. The interpretation of the data given at the very end of the listing can best be understood by recognizing that CARE III allows for redundant stages. The column of numbers under the heading, Portion of Unreliability Caused by Exhaustion of Modules of Exactly K Stages Failing by 'FT', 'time units', identifies the relative contribution K stage failures make to the total system unreliability. The greater the stage redundancy modeled, the greater the number of stage failures required to substantially increase the unreliability. In the example problem output, one stage failure contributes 2.25E-10 toward system unreliability, two stage failures contribute 8.5E-23, and three stage failures contribute 5.17E-36. One stage failure contribution

```
S U M M A R Y   I N F O R M A T I O N :

Q  SUM    =
          Ø.ØØØØØØØØØØØØE+ØØ 4.12995Ø8Ø34E-16 1.5635237178E-15 3.2956482849E-15 5.4573350233E-15
          1.Ø6Ø8219456E-14 1.6368149376E-14 2.241Ø845152E-14 2.8581886749E-14 4.1Ø66732263E-14
          5.36Ø9453844E-14 6.6164ØØ6782E-14 7.8722178244E-14 1.Ø3841563Ø1E-13 1.2896Ø9Ø174E-13
          1.54Ø8Ø29399E-13 1.7919971334E-13 2.2943851138E-13 2.7967726876E-13 3.2991599903E-13
          3.8Ø154756Ø2E-13 4.8Ø63218986E-13 5.8110948779E-13 6.8158678571E-13 7.82Ø6397521E-13
          9.8301813738E-13 1.1839717574E-12 1.3849252691E-12 1.5858782386E-12 1.9877834188E-12
          2.3896869726E-12 2.7915896591E-12 3.1934910445E-12 3.9972899123E-12 4.8Ø1Ø831422E-12
          5.6Ø487160Ø6E-12 6.4Ø8655295Ø5E-12 8.Ø162Ø66221E-12 9.6237358360E-12 1.12312468Ø5E-11
          1.2838734416E-11 1.6Ø53653198E-11 1.9268489582E-11 2.2483237494E-11 2.5697907344E-11
          3.2126995508E-11 3.8555759280E-11 4.4984193454E-11 5.1412291Ø91E-11 6.4267501Ø43E-11
          7.7121392605E-11 8.9973965778E-11 1.Ø2825227Ø5E-1Ø 1.2852378883E-1Ø 1.5421705579E-1Ø
          1.799Ø5Ø4919E-1Ø 2.Ø558782454E-1Ø 2.5693749905E-1Ø 3.Ø826613484E-1Ø 3.5957373190E-1Ø
          4.1Ø86Ø31799E-1Ø 5.1337Ø56828E-1Ø 6.1579Ø72448E-1Ø 7.1813976987E-1Ø 8.2Ø3988Ø445E-1Ø

P* SUM    =
          Ø.ØØØØØØØØØØØØE+ØØ 3.2742765169E-21 1.3Ø97118185E-2Ø 2.9468571Ø47E-2Ø 5.2388524437E-2Ø
          1.1787396107E-19 2.Ø9554Ø3Ø454E-19 3.2742796835E-19 4.7149630958E-19 8.3821478833E-19
          1.3Ø971311Ø2E-18 1.8859870995E-18 2.567Ø342627E-18 4.2434625986E-18 6.3390179742E-18
          8.8536548946E-18 1.1787408101E-17 1.8912253861E-17 2.7713528371E-17 3.8191276297E-17
          5.Ø3453156Ø2E-17 7.9682824568E-17 1.1572616759E-16 1.5847547Ø45E-16 2.Ø7930Ø2Ø376E-16
          3.2695667640E-16 4.7280643256E-16 6.4547859872E-16 8.4497402192E-16 1.3244359629E-15
          1.9111860616E-15 2.6052264357E-15 3.4Ø6564Ø730E-15 5.3310983151E-15 7.6848089Ø50E-15
          1.Ø467687372E-14 1.3679730329E-14 2.1391341Ø50E-14 3.Ø819575ØØ1E-14 4.1964495708E-14
          5.4826Ø5Ø655E-14 8.5699215736E-14 1.2343911429E-13 1.6804554981E-13 2.1951868491E-13
          3.43Ø6541982E-13 4.94Ø7841100E-13 6.7255765844E-13 8.7850516792E-13 1.3727990099E-12
          1.9769611295E-12 2.6909923942E-12 3.5148945386E-12 5.4923223088E-12 7.9092548483E-12
          1.Ø765725117E-11 1.4Ø61763Ø39E-11 2.1972694497E-11 3.1642612142E-11 4.3072143074E-11
          5.6262165Ø66E-11 8.7929Ø73744E-11 1.2665650473E-1Ø 1.7246429285E-1Ø 2.2537867406E-1Ø

Q SUM + P* SUM =
          Ø.ØØØØØØØØØØØØE+ØØ 4.1299836260E-16 1.5635368468E-15 3.2956777193E-15 5.4573875394E-15
          1.Ø6Ø8337193E-14 1.6368359440E-14 2.2411172107E-14 2.8582357699E-14 4.1Ø67569132E-14
          5.361Ø765Ø51E-14 6.6165890583E-14 7.8724746448E-14 1.Ø3845805565E-13 1.2896724433E-13
          1.54Ø8914379E-13 1.7921150404E-13 2.2945743070E-13 2.7970497012E-13 3.2995419006E-13
          3.8Ø2Ø5Ø9Ø50E-13 4.8Ø71187872E-13 5.8122522637E-13 6.8174524186E-13 7.8227192519E-13
          9.8334513275E-13 1.1844445780E-12 1.3855706946E-12 1.5867232658E-12 1.9891078801E-12
          2.3915982042E-12 2.7941947801E-12 3.1968976077E-12 4.ØØ262115Ø2E-12 4.8Ø87679672E-12
          5.6153393567E-12 6.4223348861E-12 8.Ø375983647E-12 9.6545558007E-12 1.1273211531E-11
          1.2893560351E-11 1.6139352008E-11 1.9391929035E-11 2.2651283627E-11 2.5917426191E-11
          3.247ØØ61362E-11 3.9Ø498362Ø1E-11 4.5656752684E-11 5.2290796693E-11 6.5640298752E-11
          7.9Ø98352868E-11 9.2664959039E-11 1.Ø634Ø12421E-1Ø 1.34Ø1611765E-1Ø 1.6212631238E-1Ø
          1.9Ø67Ø77Ø84E-1Ø 2.1964959Ø18E-1Ø 2.7891Ø2ØØ49E-1Ø 3.399Ø874ØØ4E-1Ø 4.Ø2645888Ø5E-1Ø
          4.6712250734E-1Ø 6.Ø129962121E-1Ø 7.4245354309E-1Ø 8.9Ø6Ø4Ø9Ø48E-1Ø 1.Ø457774646E-Ø9

K                PORTION OF UNRELIABILITY CAUSED BY        PORTION OF UNRELIABILITY CAUSED BY
STAGE            FAULT HANDLING AFTER EXACTLY K STAGES     EXHAUSTION OF MODULES OF EXACTLY K
FAILURES         HAVE FAILED BY 1.ØØØØE+Ø1 HOURS           STAGES FAILING BY 1.ØØØØE+Ø1 HOURS

     Ø                    8.2Ø39888Ø445E-1Ø                    Ø.ØØØØØØØØØØØØE+ØØ
     1                    X                                    2.2537867406E-1Ø
     2                    X                                    8.5950213711E-23
     3                    X                                    5.1767306865E-36


TOTAL SYSTEM UNRELIABILITY AT      1Ø.Ø HOURS = 1.Ø457774646E-Ø9
```

Figure 4-4. Summary Information, Example from Section 2.0

was predominate because the system fault tree was an OR gate specifying no stage redundancy. The sum of numbers for this column equals the P*SUM(FT) value, which is the last time point of the P*SUM array.

Continuing the discussion of the table shown in figure 4-4, the probability values listed under the heading, Portion of Unreliability caused by Fault-Handling after Exactly K Stages have Failed by 'FT', 'time units', identify the contribution imperfect fault-handling makes toward system unreliability as a function of the number of stage failures. For the example problem, one stage failure fails the system (OR gate for system tree) which is the reason only one nonzero value appears in the column. The 'X' in the other rows indicate a very small probability or a probability equal to zero. The sum of numbers for this middle column equals the QSUM(FT) value, which is the last time point of the QSUM array. Unlike the P*SUM array, the QSUM array can be further analyzed and broken into detail by looking at the individual $Q(t|\underline{\ell})$ arrays of the subrun output previously discussed.

The magnitude of the QSUM and P*SUM probabilities indicate the relative dependence of a system's unreliability resulting from imperfect fault-handling and due to stage failures. In the example problem discussed in this section, imperfect fault-handling makes a significant contribution to total system unreliability even though no stage failures have occurred.

5.0 REFERENCES

1. Bavuso, S. J.: A User's View of CARE III, Proceedings of the 1984 Annual Reliability and Maintainability Symposium, pp. 417-422, January 1984.

2. Stiffler, J. J.; and Bryant, L. A.: CARE III Phase II Report - Mathematical Description, NASA CR-3566, 1982.

3. Bavuso, S. J.; Brunelle, J. E.; and Petersen, P. L.: CARE III Hands-On Demonstration and Tutorial, NASA TM-85811, May 1984.

4. Trivedi, K. S.; and Geist, R. M.: A Tutorial on the CARE III Approach to Reliability Modeling, NASA CR-3488, 1981.

5. Rose, D. M.; Altschul, R. E.; Manke, J. W.; and Nelson, D. L.: Review and Verification of CARE III Mathematical Model and Code: Interim Report, NASA CR-166096, 1983.

6. Rose, D. M.; Manke, J. W.; Altschul, R. E.; and Nelson, D. L.: Correction and Improvement of CARE III, Version 3, NASA CR-166122, 1984.

7. Henley, E. J.; and Kumamoto, H.: Reliability Engineering and Risk Assessment, Prentice-Hall, 1981.

8. Moss, R. Y.: Modeling Variable Hazard Rate Life Data, Proceedings of the 28th Electronic Components Conference, pp. 16-22, April 1978.

9. Timming, A. R.: A Study of Total Space Life Performance of GSFC Spacecraft, NASA TN D-8017, 1975.

10. Shurman, M. B.: Time-Dependent Failure Rates for Jet Aircraft, Proceedings of the Annual Reliability and Maintainability Symposium, pp. 198-203, January 1978.

11. Abernethy, R. B.; Breneman, J. E.; Medlin, C. H.; and Reiman, G. L.: Weibull Analysis Handbook, Final Technical Report, July 1982 - August 1983, AFWAL TR-83-2079, AD-A 143100.

12. O'Neill, E. J.; and Halverson, J. R.: Study of Intermittent Field Hardware Failure Data in Digital Electronics, NASA CR-159268, 1980.

13. Masson, G. M.: Executive Summary - Intermittent/Transient Faults in Computer Systems, NASA CR-159229, 1979.

14. Nagel, P. M.; Scholz, F. W.; Skrivan, J. A.: Software Reliability: Additional Investigation into Modeling with Replicated Experiments, NASA CR-172378, 1984.

15. Bavuso, S. J., et al.: Latent Fault Modeling and Measurement Methodology for Application to Digital Flight Controls, Advanced Flight Control Symposium, USAF Academy, Colorado Springs, CO, August 1981.

16. McGough, J. G., et al.: Measurement of Fault Latency in a Digital Avionic Miniprocessor Part II, NASA CR-3651, 1983.

17. Stiffler, J. J., et al.: CARE III Phase III Report: Test and Evaluation, NASA CR-3631, November 1982.

18. Rice, J. W.; and McCorkle, R. D.: Digital Flight Control Reliability - Effects of Redundancy Level, Architecture and Redundancy Management Technique, AIAA paper 79-1893, August 1979.

APPENDIX A

Machine Dependency Information

It has been NASA Langley Research Center's (LaRC) goal to develop the Computer-Aided Reliability Estimation (CARE III) computer program to run on a wide range of computers. To obtain this goal, NASA LaRC has developed the second release of the CARE III computer program to conform to all ANSI Fortran 77 standards. This approach has enabled NASA to develop CARE III so that it can execute on NASA's VAX-11 700 series computers or its CYBER 170 series computers. The following pages of Appendix A describe machine dependency information for these two types of computers and should help in the installation of other machines as well.

The Digital Equipment Corporation VAX-11 700 series computers at NASA LaRC presently run under the VMS version 3.7 operating system. The VAX computers are virtual memory machines with 32 bit single precision words. Single precision floating point numbers can take on values from approximately 0.3E-38 to 1.7E+38. The VAX-11 Fortran compiler is used to compile the CARE III program for these machines.

The Control Data Corporation CYBER 170 series computers used at NASA LaRC presently run under the NOS version 1.4 operating system. These computers have a fixed core memory of 377,000 octal words, and single precision words are 60 bits long. Single precision floating point numbers can take on values from approximately 1.0E-293 to 1.0E+322. The CDC Fortran V compiler is used to compile the CARE III program on these machines.

The following discussions assume the user has successfully read the magnetic tape received from the NASA COSMIC distribution facility which contains the second release version of the CARE III program.

A.1 Installation

With the exception of the plotting routines described in Appendix A.2, the CARE III computer program consists of three main program modules. These programs are called CAREIN, COVRAGE, and CARE3. For these three program modules, there are four source code files because two slightly different CAREIN files have been supplied. The two CAREIN modules are identical except that one CAREIN module reads the user-supplied input file using ANSI Fortran 77 list-directed read statements, and the other CAREIN module uses nonstandard Fortran NAMELIST read statements for reading the user-supplied input file. The two versions of the CAREIN module were included because VAX-11 Fortran and CDC Fortran V both support the Fortran NAMELIST read statements, and because it is easier for a user to create an input file using 'namelist' syntax, than it is for him to create an input file using 'list-directed' syntax. The first few lines of each of the four Fortran source code files for the three different program modules to be used is shown below.

Once it is decided which of the two CAREIN modules is to be concatenated with the COVRGE and CARE3 modules, one can compile the complete CARE III source code. A compiler that can compile ANSI Fortran 77 as a minimum, needs to be used: A VAX-11 Fortran compiler and a CDC Fortran V compiler will compile ANSI Fortran 77 as well as the NAMELIST read extension. A chart showing how executable code is created on the VAX and CDC machines is shown in table A.1. Note the names of the pieces of code that were used at NASA LaRC and that other names can be used if desired.

Identification for CAREIN program module which uses NAMELIST statements (VAX-11 Fortran and CDC Fortran V):

```
      PROGRAM CAREIN
C*                                                                    *
C*  NOTE:  THIS CAREIN MODULE IS FOR NAMELIST FORMATTED INPUT         *
C*                                                                    *
C**********************************************************************
C*                                                                    *
C*  TITLE:   COMPUTER-AIDED RELIABILITY ESTIMATION, THIRD GENERATION *
C*  ACRONYM: CARE III.                                                *
C*                                                                    *
C**********************************************************************
C*                                                                    *
C*  CARE III, RELEASE VERSION 2.0, APRIL 1985                         *
```

Identification for CAREIN program module which uses list-directed formats for reading input (Fortran 77):

```
      PROGRAM CAREIN
C*                                                                    *
C*  NOTE:  THIS CAREIN MODULE IS FOR LIST-DIRECTED INPUT              *
C*                                                                    *
C**********************************************************************
C*                                                                    *
C*  TITLE:   COMPUTER-AIDED RELIABILITY ESTIMATION, THIRD GENERATION *
C*  ACRONYM: CARE III.                                                *
C*                                                                    *
C**********************************************************************
C*                                                                    *
C*  CARE III, RELEASE VERSION 2.0, APRIL 1985.                        *
```

Identification for COVRGE module (Fortran 77):

```
      PROGRAM COVRGE
C*                                                                    *
C**********************************************************************
C*                                                                    *
C*  TITLE:   COMPUTER-AIDED RELIABILITY ESTIMATION, THIRD GENERATION *
C*  ACRONYM: CARE III.                                                *
C*                                                                    *
C**********************************************************************
C*                                                                    *
C*  CARE III, RELEASE VERSION 2.0, APRIL 1985.                        *
```

Identification for CARE3 module (Fortran 77):

```
      PROGRAM CARE3
C*                                                                    *
C**********************************************************************
C*                                                                    *
C*  TITLE:   COMPUTER-AIDED RELIABILITY ESTIMATION, THIRD GENERATION *
C*  ACRONYM: CARE III.                                                *
C*                                                                    *
C**********************************************************************
C*                                                                    *
C*  CARE III, RELEASE VERSION 2.0, APRIL 1985.                        *
```

Compilation of VAX files

Source code	Object code	Executable code
CAREIN.FOR or CINAME.FOR	CAREIN.OBJ	CAREIN.EXE
COVRGE.FOR	COVRGE.OBJ	COVRGE.EXE
CARE3.FOR	CARE3.OBJ	CARE3.EXE

Compilation of CDC files

Source code file	Relocatable, executable binary file
CAREIN or CINAME	CRINGO5
COVRGE	COVRGO5
CARE3	CAREGO5

Table A.1

The user has a great deal of latitude in compiling the CARE III program, but here are some recommendations. Some Fortran statements can be interpreted somewhat differently from installation to installation depending upon default compiler options. For CARE III, Fortran 77 options should be chosen. As an example, "DO" loops have a minimum trip count of zero in ANSI Fortran 77. Some compilers default to a minimum trip count of one. When a computation underflows on a VAX or CDC machine the result is zero. With some types of input to the CARE III program, some of the CARE III algorithms will underflow. Therefore, if the host machine's compiler provides for checking underflow, checking for computations that underflow to zero is not recommended. The first release of the CARE III computer program required the host machine's memory to be preset to zero. For the current release version two, it is believed that all memory initialization problems have been resolved. When comparing results to the examples in this guide, the user may find that his results may only agree to about three or so significant digits. Even though the CARE III program prints many values with ten or more significant digits, many calculations are not done with that much precision. The precision of the computations is highly dependent on the host machine.

The CAREIN, COVRGE, and CARE3 program modules, which makes up most of the CARE III package, create and use several files. All files are assigned and set up using Fortran OPEN statements except for the input file used by the CAREIN module which is supplied by the user, and also with the exception of each of the program module's output. Therefore, the user must assign Fortran unit numbers to four files through the use of system commands. The user-supplied input file to the CAREIN modules will be read from Fortran unit 7, and the CAREIN output will be written to Fortran unit 6. The output to modules COVRGE and CARE3 will also be written to Fortran unit 6. By assigning the four files above using system commands, the user can more easily assign different names to the input file and output listings each time the CARE III program is run.

An example command file is shown in Appendix A.3 to execute CARE III on a VAX-11 computer operating under a VMS operating system. The comments in this command file along with figure A.1 help to show how files are used and passed among the program modules.

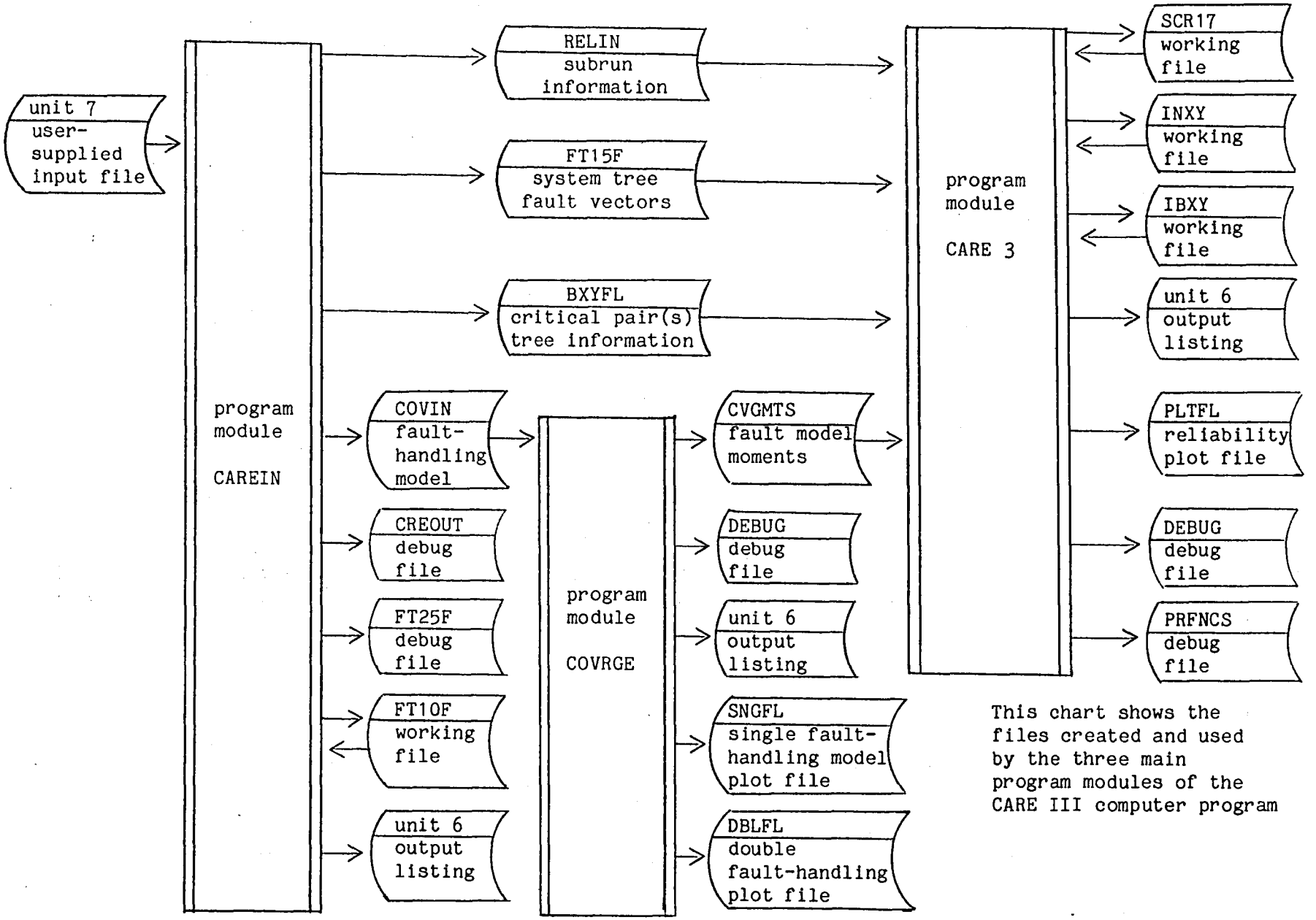


Figure A-1. CARE III File Structure

Files used by CARE III can be divided into groups. There are files used to pass information from one program module to the next, files used to debug the program when under development, files to hold plotting information, files used for memory storage (working files), and files used to hold the output information (output listings). Debug files CREOUT and FT25F are created from the CAREIN module when the user-supplied input parameter CINDBG is set .TRUE.. Debug file DEBUG is created from the COVRGE module when either variable CHIDBG or SDBUG in COVRGE's subroutine BLOCK DATA is set .TRUE. Debug files DEBUG and PRFNCS are created from the CARE3 module if either variable CHIDBG or DBGFLG in CARE3's subroutine BLOCK DATA BLK1 is set .TRUE.. All debug files are readable text files. Plot files are created by setting appropriate variables in the user-supplied input file CAREIN to .TRUE.. More information relating to plotting can be found in Appendix A.2. Working files are used by some program modules to reduce the amount of memory required by a computer without virtual memory, such as a CDC CYBER 173. The output listing to CAREIN fits within 80 columns and the output listings to COVRGE and CARE3 fit within 132 columns. The user-supplied input file used by the CAREIN module must not have records exceeding 80 columns.

To facilitate file assignment on a CDC 170 series machine, the program statements to the CAREIN, COVRGE and CARE3 program modules should be changed as follows:

```
From, PROGRAM CAREIN
To,   PROGRAM CAREIN (OUTPUT, TAPE6=OUTPUT, CREIN, TAPE7=CREIN)

From, PROGRAM COVRGE
To,   PROGRAM COVRGE (OUTPUT, TAPE6=OUTPUT)

From, PROGRAM CARE3
To,   PROGRAM CARE3 (OUTPUT, TAPE6=OUTPUT)
```

By compiling the program modules with the above changes, the NOS procedure files shown in Appendix A.4 could be used to run CARE III either interactively or batch.

As noted at the beginning of this appendix, a CDC 170 series computer works with a longer word and thus higher precision calculations than a VAX-11 700 series computer does. To take advantage of this extra precision, there are some constants in the COVRGE and CARE3 program modules which should be changed. These changes are now listed:

In program COVRGE, subroutine CMPFUN

```
From,     , REALMN = 1.0E-38
To,       , REALMN = 1.0E-293
```

In program COVRGE, subroutine VOLTRA

```
From,     ( REALMN = 1.0E-38 )
To,       ( REALMN = 1.0E-293 )
```

In program COVRGE, function PREEXP

```
From,     ( REALMX = 1.0E+38
          , REALMN = 1.0E-38
          , EXPMAX = 88.0
          , EXPMIN = -88.0

To,       ( REALMX = 1.0E+322
          , REALMN = 1.0E-293
          , EXPMAX = 741.6
          , EXPMIN = -675.8
```

In program COVRGE, subroutine HSGEAR

```
From,     UROUND = 1.1920929E-07
To,       UROUND = 7.105427357602E-15
```

In program CARE3, function PREEXP

```
From,    ( REALMX = 1.0E+38
         , EXPMAX = 88.0
         , EXPMIN = -88.0

To,      ( REALMX = 1.0E+322
         , EXPMAX = 741.6
         , EXPMIN = -675.8
```

The above constants are now defined. REALMX is the largest[1] single precision floating point number the host machine can compute with. REALMN is the smallest single precision floating point number the host machine can compute with. EXPMAX is the largest floating point number that the host's machine intrinsic, single precision, Fortran EXP function can handle without overflowing , and EXPMIN is the smallest number the EXP function will handle without underflowing. UROUND is the unit round off error of a single precision floating point number.

---

1. Some constants listed were actually set a little below their maximum values or a little above their minimum values.

A.2 Plotting Information

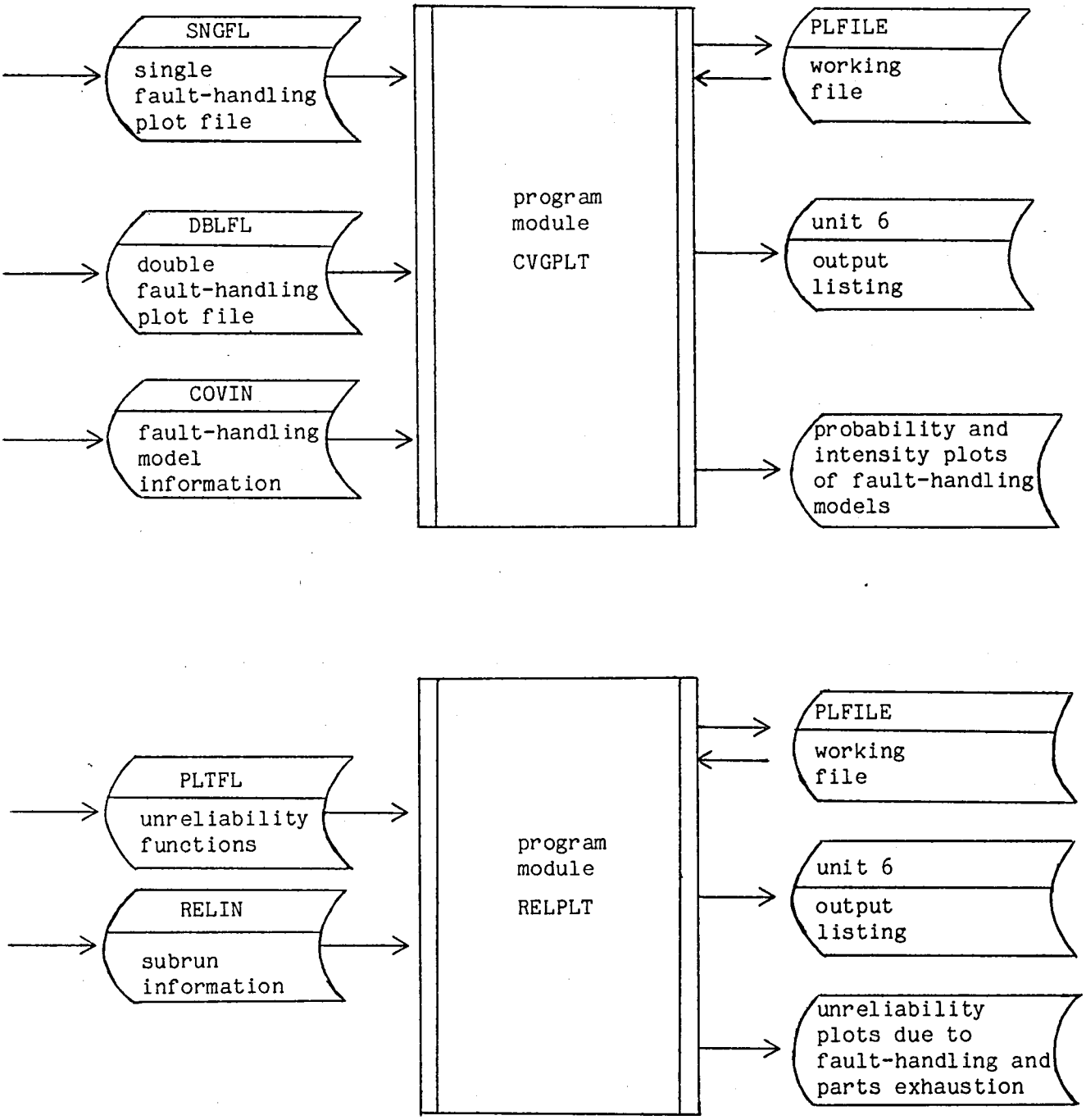
The second release version of the CARE III computer program has two optional program modules that can be used for making plots. One program module, called CVGPLT, can be used for plotting fault-handling model probability and intensity functions, and the other optional program module, called RELPLT, can be used for plotting the system unreliability summary functions. These modules are written in ANSI FORTRAN 77, and they use the DISSPLA plotting package which is a commercial product. The capability of having plotting routines using DISSPLA graphics subroutine calls was originally implemented for the Air Force Avionics Laboratory, at Wright-Patterson AFB, in 1982. NASA Langley Research Center (LaRC), however, does not have the DISSPLA plotting package and therefore has not been able to thoroughly test the two program modules. It is hoped that in the future NASA LaRC will be able to develop code using ANSI GKS graphic standards for program modules CVGPLT and RELPLT. For the present, however, the way in which the plotting modules are structured are discussed below so that users that have DISSPLA may insure the program modules work the way they should, and users who don't have it may be able to determine how to develop their own plotting capabilities.

Figure A.2 shows what files are used and produced by program modules CVGPLT and RELPLT. In relationship to the main program modules, CVGPLT can be executed after the COVRGE module is executed, and RELPLT can be executed after the CARE3 module is executed. As with any code with graphics subroutine calls, the appropriate graphics libraries or files would have to be linked or attached to CVGPLT and RELPLT. Plots produced by the original versions of CVGPLT and RELPLT can be found in reference 17.

Of the two plotting routines, the user will probably find CVGPLT to be of the least use. It was developed primarily for developmental debugging of the CARE III program. The functions which are plotted by CVGPLT are listed in section 3.1 under the 'CVPLOT' input explanation, and the math description of these functions is given in section 4.2 of reference 5.

The routine, RELPLT, is used to plot the QSUM, P*SUM, and QSUM + P*SUM arrays listed by the CARE3 module. These three unreliability summary functions are discussed in section 4.0 of this guide.

Figure A-2. Plotting File Structure



A.2.1 Data Structure of Files Used by Program Module CVGPLT

One of the files used by CVGPLT is SNGFL. Plot file SNGFL is created in program COVRGE if the general fault-handling model is chosen, by setting $FLTTYP NAMELIST input parameter 'MARKOV' = 2, and if plotting is requested, by setting input parameter 'CVPLOT' = .TRUE.

File SNGFL contains 9326 word blocks. There are 'NFTYPS' ($FLTTYP NAMELIST input parameter) blocks contained in the file for a maximum of five blocks. Each block consists of the following functions in the order listed below:

1. $P_B(t)$ - probability single fault is benign
2. $P_{\bar{B}}(t)$ - probability single fault is not benign
3. $p_F(t)$ - single fault failure intensity
4. $P_L(t)$ - probability of latent single fault
5. $p_{DP}(t)$ - single fault detected as permanent intensity.

In program CVGPLT, each block is read into the following COMMON block:

```
      COMMON/SNGFNC/  PBNG(1800), PBGSTP ,NPBGST(64)
     .               ,PNBNG(1800),PNBSTP ,NPNBST(64)
     .               ,PFLD(1800) ,PFSTEP ,NPFSTP(64)
     .               ,PLAT(1800) ,PLTSTP ,NPLTST(64)
     .               ,PDP(1800)  ,PDPSTP ,NPDPST(64) ,LNDRLG
```

The ith block is the ith fault type 'ITYP' defined in program CAREIN. 'ITYP' is used to label the fault type on the plots.

A time array 'TMAR(1800)' is generated prior to each function being plotted using the initial step size variable and step size doubling and halving array corresponding to the function currently being plotted. Function $P_B(t)$ will be used to describe the single-fault functions' plotting data structure, and the manner in which the time array is generated. Each single-fault function is stored using the same type of structure.

$P_B(t)$ is stored in array 'PBNG(IT)' using a maximum of 1800 values. Its initial step size is stored in real variable 'PBGSTP' and its step size doubling and halving information is stored in integer array 'NPBGST(ISTP)' using a maximum of 64 step size changes. The step size may change only by doubling or halving. 'NPBGST(ISTP)' defines how many of each doubled (as a

positive integer) or halved (as a negative integer) steps exist per each step size change. If 'PBGSTP' equals 0.2 and 'NPBGST(1 - 7)' = 3, 2, 4, -2, -4, 3, 2, with the remainder of the array filled with zeros, then the time array would be computed in program CVGSTP as follows:

```
TMAR( 1)  = 0.0  ┐
TMAR( 2)  = 0.2  ├      initial step size = 0.2
TMAR( 3)  = 0.4  ┘
TMAR( 4)  = 0.6  ┐
TMAR( 5)  = 1.0  ├      doubled step size = 0.4
TMAR( 6)  = 1.4  ┘
TMAR( 7)  = 2.2  ┐
TMAR( 8)  = 3.0  ├      doubled step size = 0.8
TMAR( 9)  = 3.8  ┘
TMAR(10)  = 4.6  ┐
TMAR(11)  = 5.0  ├      halved step size  = 0.4
TMAR(12)  = 5.4  ┘
TMAR(13)  = 5.6  ┐
TMAR(14)  = 5.8  ├      halved step size  = 0.2
TMAR(15)  = 6.0  ┘
TMAR(16)  = 6.2  ┐
TMAR(17)  = 6.6  ├      doubled step size = 0.4
TMAR(18)  = 7.0  ┘
TMAR(19)  = 7.4  ┐
TMAR(20)  = 8.2  ├      doubled step size = 0.8
TMAR(21)  = 9.0  ┘
```

Therefore, function 'PBNG(IT)', in this truncated example, would consist of 21 points to use for the plot, with 'IT' ranging from 1 to 21.

Input parameter 'DBLDF' (contained in $FLTTYP NAMELIST) determines the amount of points generated when the single-fault functions are computed in program COVRGE - the smaller the value given to 'DBLDF' the more points generated. Each single-fault function has a unique initial step size and step size change description. That is why the time array for each function is not contained in the plot file SNGFL. The plot file can contain a maximum of 46,630 words of plotting data, if five fault types were defined by the user. If the time arrays were included in the file, instead of the step size change information, the file would increase to a maximum of 90,005 words of plotting data. This would practically double its size and I/O access time unnecessarily.

Variable 'LNORLG', the last word contained in each block of plotting data, contains the $FLTTYP NAMELIST input parameter 'IAXSCV' defining the Y-axis scale desired for plotting the fault-handling functions. Program CVGPLT also uses file COVIN, generated by program CAREIN, to retrieve the system tree title

to help identify the current run, and the 'ITIME' variable that is used to identify a time base of 'HOURS', 'MINS ', 'SECS ', or 'MSECS'.

File DBLFL contains 1865 word blocks. There are 'NFTYPS' squared blocks contained in the file for a maximum of 25 blocks. Each block consists of the following function:

$p_{DF}(t)$ - double fault failure intensity.

In program CVGPLT, each block is read into the following COMMON block:

```
COMMON /DBLFNC/ PDFAR(1800) ,PDFSTP ,NPDFST(64)
```

The ith block is the ith double-fault type pair ('ITYP', 'JTYP'), where 'JTYP' varies the fastest. For example, if three fault types had been defined in program CAREIN, the double-fault failure intensity function, computed per fault type pair, would be written to DBLFL ordered (1,1), (1,2), (1,3), (2,1), (2,2), (2,3), (3,1), (3,2), and (3,3).

A time array 'TMAR(1800)' is generated prior to each double-fault function being plotted using the initial step size variable `PDFSTP' and the step size change description 'NPDFST(64)'. The time array is generated in the exact same manner as described above for the single-fault functions.

The plot file DBLFL can contain a maximum of 46,625 words of plotting data, if five fault types were defined. In general there are fewer double-fault functions to plot than single-fault functions because there are five times 'NFTYPS' single-fault functions and 'NFTYPS' squared double-fault functions, in files SNGFL and DBLFL respectively. At its maximum size, it would contain 90,000 words of plotting data if the time arrays were passed in the plotting file. DBLFL does not contain the 'IAXSCV' input parameter. Program CVGPLT uses variable 'LNORLG' from file SNGFL for the choice of the Y-axis scale.

A.2.2 Data Structure of Files Used By Program Module RELPLT

Plot file PLTFL is created in program CARE3, if input parameter 'RLPLOT' is set .TRUE. in $STAGES NAMELIST. File PLTFL contains 260 words of plotting data that consists of one 195 word block and one 65 word block. The first block contains the CARE3 summary results: QSUM, P*SUM, and QSUM+P*SUM read by program RELPLT into COMMON /PLTCOM/ QLTSUM(65) ,PSTSUM(65) ,QPSTSM(65). The second block contains the time values corresponding to the summary functions' results and is read into array 'TMAR(65)' contained in the following COMMON block:

```
      COMMON /STEPCM/  ITSTPS    , MAXSTP   , RELSTP   , ITIME
     .                , TMAR(65) , NSTGRN   , KWT      , PSTRNC
     .                , IPRINT   , RLPLOT   , IAXSRL   , QPTRNC
     .                , CPLFLG   , SYSMNT   , TBCF
C
      LOGICAL            RLPLOT   , SYSMNT   , CPLFLG
```

Due to the unequal step sizes used in computing the enhanced CARE3 functions, if input parameter 'LFTMST' was set .TRUE. in $FLTTYP NAMELIST, the functions' corresponding time values are written to file PLTFL, and hence no longer generated in program RELPLT. This increases the size of file PLTFL by only 65 words since all three functions were computed using the same step sizes.

Program RELPLT also uses file 'RELIN', generated by program CAREIN, to retrieve the system tree title to help identify the current run, the ITIME variable used to specify the time base of the plots, and input parameter 'IAXSRL' to determine the Y-axis plotting choice.

A.3

```
        *** COMMAND FILE TO RUN THE CARE-III COMPUTER PROGRAM ***
$ !
$ !
$ !  TO RUN CARE-III, EXECUTE THIS COMMAND FILE WITH THE CARE-III INPUT
$ !  FILE AS THE ONLY PARAMETER (i.e. @RUNCARE filename).  AFTER EXECUTION
$ !  THE INPUT FILE AND THE OUTPUT LISTING TO THE PROGRAM WILL BE LOCATED
$ !  ON FILE CAREIII.OUT.
$ !
$ !  IF YOU DESIRE PLOT, DEBUG, OR WORKING FILES CREATED BY CARE-III YOU
$ !  WILL HAVE TO ELIMINATE THE DELETE COMMAND FOR THAT FILE.  SOME OF
$ !  THE FILES NAMED IN DELETE COMMANDS MAY NOT EXIST DEPENDING UPON THE
$ !  TYPE OF DATA THAT IS IN THE INPUT FILE AND HENCE A SYSTEM ERROR MAY
$ !  RESULT WHEN THIS COMMAND FILE TRIES TO DELETE IT.  Example - PLTFL.DAT
$ !  DOES NOT EXIST UNLESS THE RELIABILITY PRINT FLAG 'RLPLOT' IS TRUE.
$ !
$ !
$ !                    **EXECUTION OF THE CAREIN MODULE**
$ !
$ !  THE CAREIN PROGRAM PROCESSES THE INPUT FILE WHICH AS MENTIONED BEFORE
$ !  IS THE ONLY PARAMETER TO THIS COMMAND FILE.
$ !
$ !  THE CAREIN PROGRAM PRODUCES SEVERAL OUTPUT FILES.  COVIN.DAT IS USED
$ !  BY THE COVRGE PROGRAM.  RELIN.DAT, BXYFL.DAT, AND FT15F.DAT ARE USED
$ !  BY THE CARE3 PROGRAM.  CREOUT.DAT AND FT25F.DAT ARE DEBUG FILES
$ !  CREATED WHEN 'CINDBG' IS SET TRUE.  FILE FT10F.DAT IS A WORKING FILE
$ !  USED FOR FAULT TREE PROCESSING.  FILE CAREIN.DAT IS THE OUTPUT
$ !  LISTING PRODUCED BY PROGRAM CAREIN.
$ !
$ASSIGN 'P1'       FOR007
$ASSIGN CAREIN.DAT FOR006
$ !
$RUN  CAREIN
$ !
$DELETE CREOUT.DAT;*
$DELETE FT25F.DAT;*
$DELETE FT10F.DAT;*
$ !
$ !
$ !                    **EXECUTION OF THE COVRGE MODULE**
$ !
$ !  THE ONLY INPUT FILE USED BY PROGRAM COVRGE IS COVIN.DAT.  OUTPUT
$ !  FILE CVGMTS.DAT IS USED BY THE CARE3 PROGRAM MODULE.  OUTPUT FILES
$ !  SNGFL.DAT AND DBLFL.DAT ARE CREATED WHEN INPUT PARMATER 'CVPLOT' IS
$ !  SET TRUE.  DEBUG FILE DBUG.DAT IS CREATED IF EITHER OF THE "HARD-WIRED"
$ !  PARAMETERS 'CHIDBG' OR 'SDBUG' IN THE PROGRAM'S BLOCKDATA SUBROUTINE
$ !  IS SET TRUE.  COVRGE.DAT IS THE OUTPUT LISTING PRODUCED BY PROGRAM
$ !  COVRGE.
$ !
$ASSIGN COVRGE.DAT FOR006
$ !
$RUN  COVRGE
$ !
$DELETE COVIN.DAT;*
```

```
$DELETE SNGFL.DAT;*
$DELETE DBLFL.DAT;*
$ ! DELETE DBUG.DAT;*
$ !
$ !
$ !                    **EXECUTION OF THE CARE3 MODULE**
$ !
$ !  THE INPUT FILES USED BY CARE3 ARE RELIN.DAT, BXYFL.DAT, FT15F.DAT
$ !  CREATED BY CAREIN, AND FILE CVGMTS.DAT CREATED BY COVRGE.  WORKING
$ !  FILES USED BY CARE3 ARE SCR17.DAT, INXY.DAT, AND IBXY.DAT.  THE
$ !  OUTPUT FILE PLTFL.DAT IS CREATED WHEN THE INPUT PARAMETER 'RLPLOT'
$ !  IS SET TRUE.  OUTPUT FILES DBUG.DAT AND PRFNCS.DAT ARE CREATED IF
$ !  EITHER OF THE "HARD-WIRED" PARAMETERS 'CHIDBG' OR 'DBGFLG' IS SET
$ !  TRUE IN CARE3'S BLOCKDATA SUBROUTINE.  CARE3.DAT IS THE OUTPUT
$ !  LISTING PRODUCED BY CARE3.
$ !
$ASSIGN CARE3.DAT  FOR006
$ !
$RUN  CARE3
$ !
$DELETE RELIN.DAT;*
$DELETE BXYFL.DAT;*
$DELETE FT15F.DAT;*
$DELETE CVGMTS.DAT;*
$DELETE SCR17.DAT;*
$DELETE INXY.DAT;*
$DELETE IBXY.DAT;*
$DELETE PLTFL.DAT;*
$ ! DELETE DBUG.DAT;*
$ ! DELETE PRFNCS.DAT;*
$ !
$ !
$ !     CREATE OUTPUT FILE CAREIII.OUT
$ !
$COPY CAREIN.DAT,COVRGE.DAT,CARE3.DAT CAREIII.OUT
$ !
$DELETE CAREIN.DAT;*
$DELETE COVRGE.DAT;*
$DELETE CARE3.DAT;*
```

A.4 Below are two NOS procedure files for running the CARE III program on a CDC 170 series computer.

A.4.1 Example procedure file for executing CARE III interactively

```
.PROC,RUNCARE,I=CREIN.
IFE,$I$=$$,LB10.
   REVERT. #I MUST BE SPECIFIED.
ENDIF,LB10.
IFE,.NOT.FILE(I,AS),LB20.
   GET,I.
ELSE,LB20.
   REWIND,I.
ENDIF,LB20.
MAP,OFF.
CRINGO5,,,I.
COVRGO5.
CAREGO5.
REVERT.
```

A.4.2 Example Procedure File for Executing CARE III as a Batch Job

```
.PROC,C3V5,I=CREIN.
IFE,$I$=$$,LB1.
   REVERT.#1 MUST BE SPECIFIED.
ENDIF,LB1.
DELIVER. BIN15PP. PAUL PETERSEN. B1220.
SEND,C5JOB,M=T,DC=IN.
REVERT. CARE3 VERSION 5 JOB SUBMITTED TO T.
.DATA,C5JOB.
SUBJOB,T100.
USER,123456C,PASWORD.
CHARGE,654321,LRC.
DELIVER. BIN15PP. PAUL PETERSEN. B1220
GET,I,CRINGO5,COVRGO5,CAREGO5.
REWIND,I.
MAP,OFF.
CRINGO5,OUTPUT,,I.
COVRGO5,OUTPUT.
CAREGO5,OUTPUT.
```

A.5 Error and Warning Messages

Error and warning messages are listed in this section. The messages are often succinct or cryptic and may cause the user some frustration. They are nevertheless included here because it was felt that even more frustration would result otherwise. Error and warning messages for subprogram CAREIN are not included; however, because these messages were enhanced. Further improvements are forthcoming.

Error Messages

The following sections contain the error and warning messages generated by modules COVRGE and CARE3 (messages for module CAREIN are not listed in this revision). The utility routine error messages are fatal to the computer run; therefore, the run will halt after one occurs. The warning messages are nonfatal and are used to alert the user to any out-of-the-ordinary processsing.

The variable enclosed in single quote marks are used to show which variables will have their contents printed. The following variables, used in the error messages, have values which do not change during the course of the computer run:

Variable Name	Variable	Description
MAXTYP	5	Maximum Number of Fault Types
MXSTGS	70	Maximum Number of Stages
MAXCAT	5	Maximum Number of Fault Categories
MXUNTS	70	Maximum Number of Coupled Units Per Fault Tree
MXCPLS	20	Maximum Number of Coupled Stages Per Fault Tree
MXSBRN	35	Maximum Number of Subruns
INMAX	513	Maximum Index Into Coverage Functions
MXSTGF	13	Maximum Number of Stage Failures
IJMAX	210	Maximum Number of Unique Fault Pair Functions Per BXYFL Subfile
MAXSTG	20	Maximum Number of Stages Per Subrun

Beside each message is a code used to differentiate the user errors from the possible internal program errors for which tests are made.

COVRGE ERROR MESSAGES (FATAL)

** ERROR - NUMBER OF FAULT TYPES REQUESTED = 'NTYPS' BUT MUST BE LESS THAN OR EQUAL TO 'MAXTYP'.	IE
** ERROR - INCORRECT RELIABILITY TIME BASE = 'TBASE'.	IE
** ERROR - THRASHING OCCURRED IN SUBROUTINE COMPFUN. RERUN PROGRAM WITH A DIFFERENT 'DBLDF' VALUE.	
** ERROR - ITH G SINGLE-FAULT FUNCTION = 'ITH' WHILE ONLY G FUNCTIONS 1 THROUGH 9 MAY BE COMPUTED WITH FUNCTION FGSNGL.	IE
** ERROR - 'DELTA'('ITYP') EQUALS ZERO, WHILE THE CONSTANT DENSITY FUNCTION WAS CHOSEN FOR 'DELTA'. EITHER ASSIGN 'DELTA' A VALUE OR SPECIFY CONSTANT RATE FUNCTION FOR THIS FAULT TYPE.	UE
** ERROR - 'RHO'('ITYP') EQUALS ZERO, WHILE THE CONSTANT DENSITY FUNCTION WAS CHOSEN FOR 'RHO'. EITHER ASSIGN 'RHO' A VALUE OR SPECIFY CONSTANT RATE FUNCTION FOR THIS FAULT TYPE.	UE
** ERROR - BOTH 'RHO' AND 'DELTA' ('ITYP') EQUAL ZERO, WHILE THE CONSTANT DENSITY FUNCTION WAS CHOSEN FOR BOTH FAULT RATES. EITHER ASSIGN THEM A VALUE OR SPECIFY CONSTANT RATE FUNCTIONS FOR THIS FAULT TYPE.	UE
** ERROR - ITH C DOUBLE-FAULT FUNCTION = 'ITH' WHILE ONLY C FUNCTIONS 1 AND 2 MAY BE COMPUTED WITH FUNCTION FCDBL.	IE
** ERROR - ITH F DOUBLE-FAULT FUNCTION = 'ITH' WHILE ONLY F FUNCTIONS 1 AND 2 MAY BE COMPUTED WITH FUNCTION FFDBL.	IE

NOTE: UE = USER ERROR; IE = INTERNAL ERROR.

COVRGE ERROR MESSAGES (FATAL)

** ERROR - ITH B DOUBLE-FAULT FUNCTION = 'ITH' WHILE ONLY B FUNCTIONS 1 AND 2 MAY BE COMPUTED WITH FUNCTION FBDBL. IE

** ERROR - THRASHING OCCURRED IN SUBROUTINE SUMARS. RERUN PROGRAM WITH A DIFFERENT 'DBLDF' VALUE. UE

** ERROR - THRASHING OCCURRED IN SUBROUTINE VOLTERA. RERUN PROGRAM WITH A DIFFERENT 'DBLDF' VALUE. UE

** ERROR - THRASHING OCCURRED IN SUBROUTINE VLTNREC. RERUN PROGRAM WITH A DIFFERENT 'DBLDF' VALUE. UE

** ERROR - INDEX PASSED TO FUNCTION FTCHSTP = 'INDX' IS LARGER THAN THE TOTAL NUMBER OF POINTS COMPUTED. IE

** ERROR - ILLEGAL MCHI OF 'MCHI' WHICH SPECIFIES WHICH SINGLE OR DOUBLE-FAULT FUNCTION TO STORE INTO COMMON /CVRGCOM/, MUST BE IN THE RANGE OF 1 THROUGH 6 ONLY. IE

** ERROR WITH MOMNT = 'MOMNT' IN FUNCTION SIMPINT. ONLY MONENTS 0,1, AND 2 ARE VALID. IE

** ERROR WITH INDICES REPRESENTING INTEGRATION LIMITS IN FUNCTION SIMPINT: 'ITFROM' = 'ITFROM' 'ITTO' = 'ITTO' WHERE 'INMAX' = 'INMAX'. IE

** ERROR - INDEX INTO 'TMAR' = 'IT' IS GREATER THAN MAXIMUM ALLOWABLE INDEX OF 'INMAX'. IE

** ERROR - CUMULATIVE ERROR OUT OF ACCEPTABLE BOUNDS; ERRONEOUS RESULTS. RERUN PROGRAM WITH A SMALLER 'DBLDF' VALUE. IE

NOTE: UE = USER ERROR; IE = INTERNAL ERROR.

COVRGE WARNING MESSAGES (NON-FATAL)

** WARNING - NSTPIN GREATER THAN 'NSTPMX' IN SUBROUTINE COMPFUN. ZERODF DECREASED TO 'ZERODF'.

** WARNING - NUMBER OF POINTS REQUIRED TO DEFINE A FUNCTION ARRAY IS LARGER THAN 'INMAX'. ZERODF INCREASED TO 'ZERODF'.

** WARNING - NSTPIN GREATER THAN 'NSTPMX' IN SUBROUTINE SUMARS. ZERODF DECREASED TO 'ZERODF'.

** WARNING - NPSTPIN GREATER THAN 'NSTPMX'. ZERODF DECREASED TO 'ZERODF'.

** WARNING - NUMBER OF POINTS REQUIRED TO DEFINE P ARRAY IS LARGER THAN 'INMAX'. ZERODF INCREASED TO 'ZERODF'.

CARE3 ERROR MESSAGES (FATAL)

** ERROR - INCORRECT TIME BASE = 'TBASE'.	IE
** ERROR - 'KWT' = 'KWT'. 'KWT' MUST BE INPUT WHEN THE SYSTEM MINTERM FILE 'FT15F' IS MISSING. THE NUMBER OF STAGE FAILURES CANNOT EXCEED 'MXSTGF'.	UE
** ERROR - SUBRUN NUMBER EXCEEDED MAXIMUM OF 'MXSBRN'.	IE
** ERROR - EOF ENCOUNTERED ON UNIT 4 DURING BUFFER IN.	IE
** ERROR - WITH SYSTEM MINTERM FILE 'FT15F' - 'PRBMT' (MNTRMV(ISTG),ISTG=1,NSTGRN)	IE
** ERROR - NUMBER OF UNIQUE FAULT PAIR FUNCTIONS = 'NBXY' FOR 'NSTGS' COUPLED STAGES, WHICH EXCEEDS THE MAXIMUM OF 'IJMAX' FOR THE MAXIMUM NUMBER OF COUPLED STAGES = 'MAXSTG'.	IE
** ERROR - INVALID BXYAR INDEX OF 'INBXY' EXISTS IN INDEX ARRAY IJSTGIN('IJSTG').	IE
** ERROR - INVALID CONDITIONAL FAULT VALUE OF 'LC' EXISTS IN CONDITIONAL FAULT VECTOR LCNDVEC('I').	UE
** ERROR - INVALID SPECIFICATION OF WHICH BXY FUNCTION DEFINITION RECORD IS CURRENTLY IN MEMORY - IBREC = 'IBREC'.	IE
** ERROR - EOF ENCOUNTERED ON PREVIOUS BUFFER IN ON UNIT 'IUNIT' WHILE TRYING TO READ RECORD 'IT'.	IE

NOTE: UE = USER ERROR; IE = INTERNAL ERROR.

** ERROR - PARITY ERROR ENCOUNTERED DURING BUFFER IN ON UNIT 'IUNIT' WHILE TRYING TO READ RECORD 'IT'.	IE
** ERROR - EOF ENCOUNTERED ON PREVIOUS BUFFER OUT ON UNIT 'IUNIT' WHILE TRYING TO WRITE RECORD 'ITB'.	IE
** ERROR - PARITY ERROR ENCOUNTERED DURING BUFFER OUT ON UNIT 'IUNIT' WHILE TRYING TO WRITE RECORD 'ITB'.	IE
** ERROR - INVALID SPECIFICATION OF WHICH BXY FUNCTION DEFINITION RECORD IS CURRENTLY IN MEMORY - IBREC = 'IBREC'.	IE
** ERROR - CURRENT BLOCK OF CRITICAL PAIR DATA IN MEMORY, STARTING AT STAGE 'KFSTG' DOES NOT MATCH THE CURRENT SUBRUN STARTING AT STAGE 'IFSTG'.	IE
** ERROR IN FPMUX -'MUX' LATENT FAULTS IN STAGE 'ISTG' IS LARGER THAN THE TOTAL NUMBER OF FAULTS = 'LX'.	IE
** ERROR - FGST CALLED WITH AN INVALID MCHI OF 'MCHI'.	IE
** ERROR - INCORRECT FAILED STATE REQUESTED FOR USE IN FUNCTION FHSFST: MCHI = 'MCHI'.	IE
** ERROR - ILLEGAL FAULT TYPE = 'ITYP' WHILE TRYING TO READ COVERAGE FUNCTIONS IN ROUTINE FHSFST.	IE
** ERROR - INCORRECT FAILED STATE REQUESTED FOR USE IN FUNCTION FHDFST: MCHI = 'MCHI'.	IE

NOTE: UE = USER ERROR; IE = INTERNAL ERROR.

CARE3 ERROR MESSAGES (FATAL)

** ERROR - ILLEGAL FAULT TYPE(S) = 'ITYP' AND/OR 'JTYP' WHILE TRYING TO READ COVERAGE FUNCTIONS IN ROUTINE FHDFST. IE

** ERROR - INCORRECT FAILED STATE REQUESTED FOR USE IN FUNCTION FFSFST: MCHI = 'MCHI'. IE

** ERROR - INCORRECT FAILED STATE REQUESTED FOR USE IN FUNCTION FFDFST: MCHI = 'MCHI'. IE

** ERROR - ILLEGAL NUMBER PASSED TO N FACTORIAL FUNCTION = 'N'. IE

CARE3 WARNING MESSAGES (NON-FATAL)

** WARNING - SYSTEM MINTERM FILE 'FT15F', GENERATED BY PROGRAM CAREIN, IS EMPTY. CARE3 WILL GENERATE P*'S WITHOUT USING THE SYSTEM FAULT-TREE DEFINITION, SINCE A 'KWT' VALUE OF 'KWT' WAS SPECIFIED.

** WARNING - CRITICAL FAULT PAIR FILE 'BXYFL' IS EMPTY. THIS RUN ASSUMES THAT NO CRITICAL PAIRS OF FAULTS EXIST.

NOTE: UE = USER ERROR; IE = INTERNAL ERROR.

UTILITY ROUTINE ERROR MESSAGES (FATAL)

** ERROR - ILLEGAL COMBINATORIAL; NFAC = 'NFAC' KFAC = 'KFAC'.	IE
** ERROR - ILLEGAL VALUE FOR 'INOUT' IN SUBROUTINE BUFBLK.	IE
** ERROR - EOF ENCOUNTERED ON UNIT 'IUNIT' DURING BUFFER 'BTYPE'.	IE
** ERROR - PARITY ERROR ENCOUNTERED ON UNIT 'IUNIT' DURING BUFFER 'BTYPE'.	IE

NOTE: UE = USER ERROR; IE = INTERNAL ERROR.

APPENDIX B

Module Status and Dependency

B.1 Module Status

Modules within a stage are in one of three disjoint states: in-use, spare, and deleted from the system. An in-use module is a functioning part of the system. A spare module is available to the system to replace an identical module which has been deleted. A deleted module has been identified (possibly incorrectly) as faulty and isolated from the system. Spares are assumed to have identical fault-handling characteristics as in-use module for a given stage. Some system configuration schemes which make use of this assumption are ones in which self fault detection programs are run on the spares and/or ones in which spares are frequently being swapped with in-use modules. Unmonitored spares cannot be directly modeled using CARE III.

In computing operational, $P(t|\underline{\ell})$, and fault-handling failure, $Q(t|\underline{\ell})$, probabilities, the CARE III computer program uses a bookkeeping method that keeps track of functional use of modules versus keeping track of the physical modules themselves. The user becomes involved in this functional bookkeeping scheme for modules when he specifies the N, M, and NOP parameters of the STAGES input paragraph (see section 3.2), and when specifying one or more critical-pair trees.

When specifying an N(x) parameter, the user is indicating that all N modules for stage x are all in working order when the system under study is first checked out or turned on. The M(x) input parameter indicates the minimum number of any physical combination of M modules that will allow stage x to perform its function. The NOP(j,x) parameter allows the user to model systems in which not all available modules are in-use at a given time. As the pool of available modules for stage x is reduced from N(x) to M(x) through fault occurrences and successful fault handling, the number of in-use and spare modules may change.

As an example of what can be modeled, consider a computer stage x which consists of seven computer modules configured into two working triads and a spare. Let's further assume that the system under study only needs the computational power of one triad, and that any two of the original seven

modules can keep this needed triad working (fig. B-1). For this problem, we would choose N(x) = 7, M(x) = 2, NOP(1,x) = 6, NOP(2,x) = 3, and we would leave NOP(3,x), NOP(4,x), and NOP(5,x) at their default value of zero. If functional module number two fails in stage x, it will be replaced by module seven, the spare, which then functionally becomes module two. If a second failure occurs, say module three fails, then module six, the next largest numbered module, assumes its role and number. At this time, modules four and five become spares and the surviving stages operate as a single triad with two spares. The CARE III program computes this because five is less than the value of NOP(1,x); thus NOP(2,x) or three modules are assumed to be in-use.

When the user selects the default value of zero for a NOP value, the CARE III program will internally calculate the NOP value which assumes that all available modules are in-use. If all NOP values had been selected to equal their default value of zero for the example, the CARE III program would have internally computed NOP(1,x) = 7, NOP(2,x) = 6, NOP(3,x) = 5, NOP(4,x) = 4, and NOP(5,x) = 3. Just like before, as modules become faulty, the largest numbered non-deleted module assumes the function and number of the deleted module. For the original example shown in fig. B-1, the CARE III program calculates NOP(3,x) = 2. Since the internally calculated value of NOP(3,x) equals M(x) in the example, the program realizes NOP(4,x) and NOP(5,x) to be irrelevant.

The whole importance of the NOP parameters is in how $Q(t|\underline{\ell})$ values are computed with regard to critical-pair tree(s). If no critical-pair tree(s) for a CARE III program is given, then the CARE III program will not use any of the NOP parameter data. With respect to $Q(t|\underline{\ell})$ computations, NOP limits the number of critical-pair failures that are associated with a particular $\underline{\ell}$ vector, or particular operating state of the system under study. CARE III assumes spares cannot contribute to critical-pair failures. For the example shown in figure B.1, functional module seven will never be able to contribute to a critical-pair failure. After two module failures, functional module number four will not be able to contribute to a critical-pair failure because it becomes a spare. Thus without the NOP information, some $Q(t|\underline{\ell})$ values would reflect overly conservative probabilities of failure due to improper fault handling.

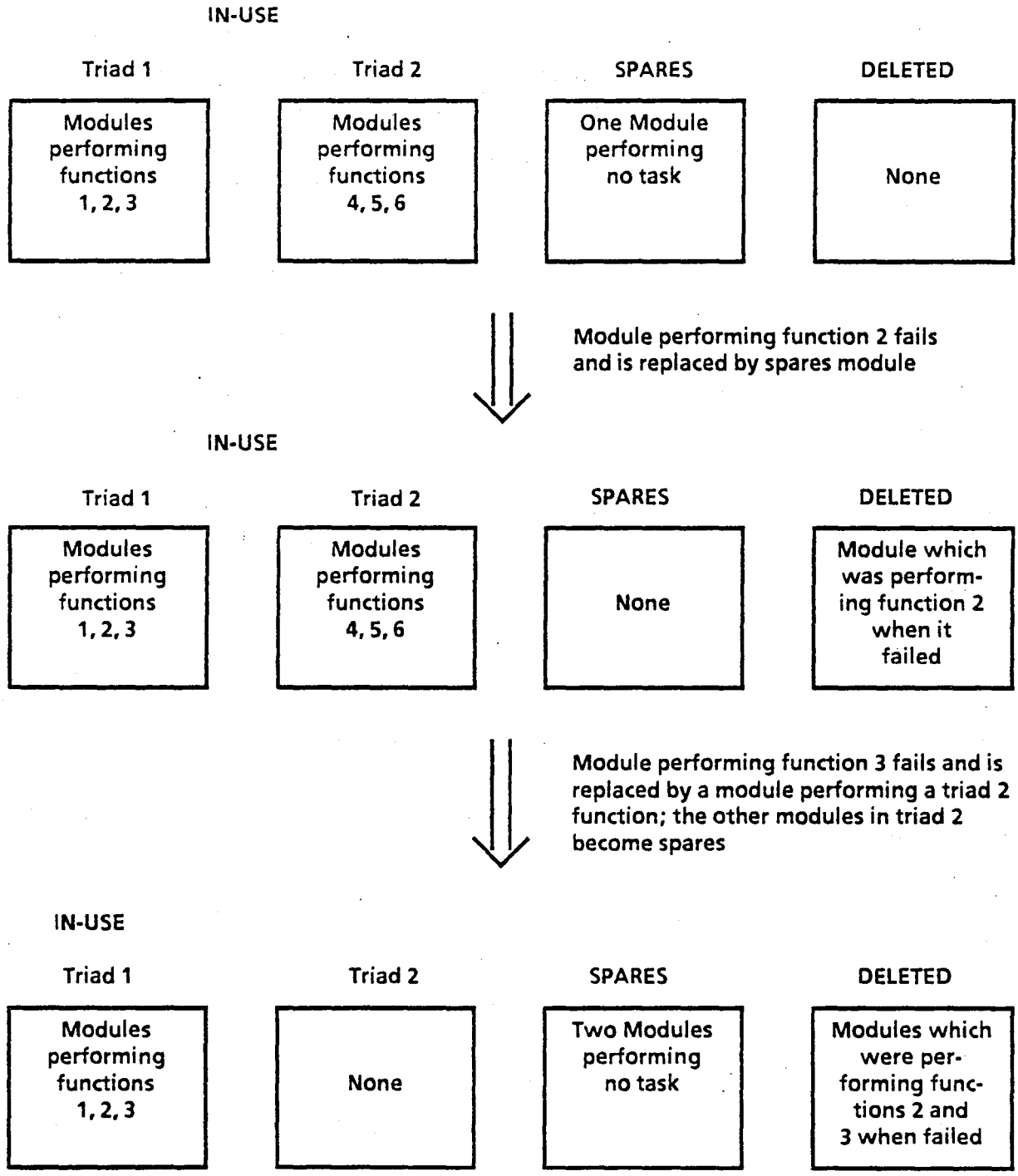


Figure B-1 Replacement Scheme Implemented in CARE III
N=7, NOP = (6,3)

Just as functional numbering applies to the NOP parameters, the user must also observe that functional numbering is applied to module numbers associated with a critical-pair tree. To say that modules five and six are critically coupled in the example shown in figure B-1, is much different than saying modules one and two are. If modules five and six were critically coupled, they would be less likely to cause a critical-pair failure than if modules one and two were assigned to be critically coupled. The reason for this is if during a mission the system under study loses and recovers from the loss of any two modules from stage x, functional module six will no longer be around to add to a critical-pair failure. Functional module one, however, will be the last module to be deleted from stage x and thus will have more time to experience a fault-handling failure.

A single fault system failure probability is computed by CARE III when the parameter C is assigned a value of less than one in the single fault-handling model. A single-fault system failure is a fault, which by itself, causes system failure when improperly handled. It needs to be noted at this time that the NOP parameters and any critical-pair trees, if specified, will not affect the way CARE III calculates the probability of system failure due to single faults. This means CARE III assumes that spares as well as in-use modules contribute to system failure when the stage that contains them is linked to a fault-handling model whose 'C' parameter is set less than one. Thus, spares are treated completely different with regard to single faults that cause a single-fault failure than compared to critically-paired faults that cause a critical-pair failure.

B.2 Time Sequential Dependence

A CARE III user may want to model a system architecture which may include time sequential dependencies among modules. By time sequential dependency it is meant that the order in which modules fail and are isolated from a system makes a difference as to which operational state the system is in. For example, suppose a fault-tolerant multiprocessor system has an LRU (line replaceable unit) that contains a power supply, a clock, a processor, a memory, and two bus guardian units. If the clock fails and is correctly isolated, the entire LRU is removed from the system. Thus the number of operational memories, processors, etc., will be reduced. If the associated processor had already failed and been isolated, however, then the clock failure would not reduce the number of processors. To keep track of modules by physical identification, the functional numbering algorithm implemented in CARE III is not entirely satisfactory. Failure to allow for time sequential dependencies may lead to an underestimate of module depletion unreliability (P*SUM calculation). In order for the CARE III program to keep track of physical identification of modules which is needed to model time sequential dependencies, the user has to specify every module in his problem to be a stage. In doing this, however, the NOP parameters can no longer be used to model spares. Furthermore, the complexity of specifying a system with each module being a stage can be overwhelming. A moderate sized system composed of seven stages with ten modules per stage could require thousands of lines to describe the dependence.

An approximate solution for modeling time sequential dependency is to use the CARE III program to calculate fault-handling unreliability, QSUM, and to calculate module depletion unreliability, P*SUM, separately. To calculate the fault-handling unreliability, one could model all sparing and fault-handling fully, ignoring the dependence. Although the value of the fault-handling unreliability, QSUM, cannot in general be guaranteed to be conservative, it should provide a reasonable estimate for many cases. A conservative bound could be obtained for the QSUM calculation, however, by representing all pairs of faults between modules to be critically coupled. This approach leads to a conservative result for QSUM since some pairs of faults will be assumed critically coupled when they are not.

To calculate the depletion failure probability for the above scheme, a separate CARE III run would be done. This time the time sequential dependency would be

fully represented letting each module be a stage. If the system can be represented without a user input error (not a trivial task) and the number of stages is within CARE III restrictions (N≤70), the depletion failure probability will be estimated by P*SUM.

A variation of the above approach that often times can lead to a significant reduction in computational time is based on the observation that for ultrareliable systems the fault-handling unreliability, QSUM, is dominant. The P*SUM computation can often be conservatively estimated by assuming that module failures are stochastically independent, e.g., in the example problem discussed above, all the modules within an LRU can be lumped together to form a single stage. The combined stage failure rate is the sum of the module failure rates so that any module failure causes stage failure. This model produces a conservatively high module failure probability compared to the model that properly treats time sequential dependency. If the P*SUM values are lower in value than the QSUM values, then a more refined fault-occurrence model is unnecessary.

B.3 Common Mode Events

An event which causes multiple basic events is called a common cause (ref. 7). The events caused by a common cause are called common mode events (CME) of the cause. In CARE III terminology, the basic events are stage failures (module failures if stages are so defined). CME's are present when an event (stage ID) appears in more than one place at the lowest level of the system fault tree (see section 3.5), or when the output to an event in the system fault tree feeds two or more upper level events. For example, a stage with ID number 5 could appear as an input to more than one gate. As another example, the output of a gate numbered 93 could feed into gates numbered 94 and 101. The effect that CME's produce is the loss of two more more subsystems given a single stage or event failure. CME's differ from single-point failures in that single-point failures cause the entire system to fail whereas a CME may or may not fail the entire system. The structure of the system tree makes that determination and is a function of stage redundancy.

APPENDIX C

General Double Fault Handling Model

The concept of critically coupled system failures is a relatively recent addition to the reliability modeling of fault-tolerant systems. For this reason, a brief tutorial follows as it is essential that the user understands the model.

With the inclusion of a double fault handling model, CARE III captures the notion of critically coupled system failures. Highly reliable fault-tolerant systems are commonly designed without single-fault system failure mechanisms, or, perhaps more realistically, with an extremely low probability of single-fault failure occurrence. Often, the dominant system failure cause is a critically coupled double fault. In these systems, most double faults are tolerated by the system; however, certain critical groupings of double faults are often not tolerated by the system and will cause system failure. It is the aim of these system designs to make critically coupled faults less probable than the mission desired unreliability. An illustrative example of a critically coupled system failure is the occurrence of two faults, one each in two of three different voting or comparison-monitoring computers performing the same flight-crucial control computations. A fault in a spare computer and a fault in the voting triad, however, would not be critically coupled.

The user specification of the critically coupled faults is accomplished by the user defining a critical-pair tree. The critical-pair tree defines all combinations of critically coupled modules, i.e, all pairs of modules that will cause system failure when faults exist in the two modules. The critical-pair tree for a voting triad is simply a 2 out of 3 logic gate. In more practical systems, the critical-pair tree can become quite complex, as critical-pair faults may occur not only within a stage (such as the voting triad) but also across stages. An example of the latter application would be a critically coupled failure between a computer module and a computer bus module as illustrated in appendix G.2, example problem 7. Since three voting computers communicate over three buses, a failure in a voting triad and a failure in a bus not connected to the failed computer would preclude a correct majority vote at the receiving end of the triplex bus.

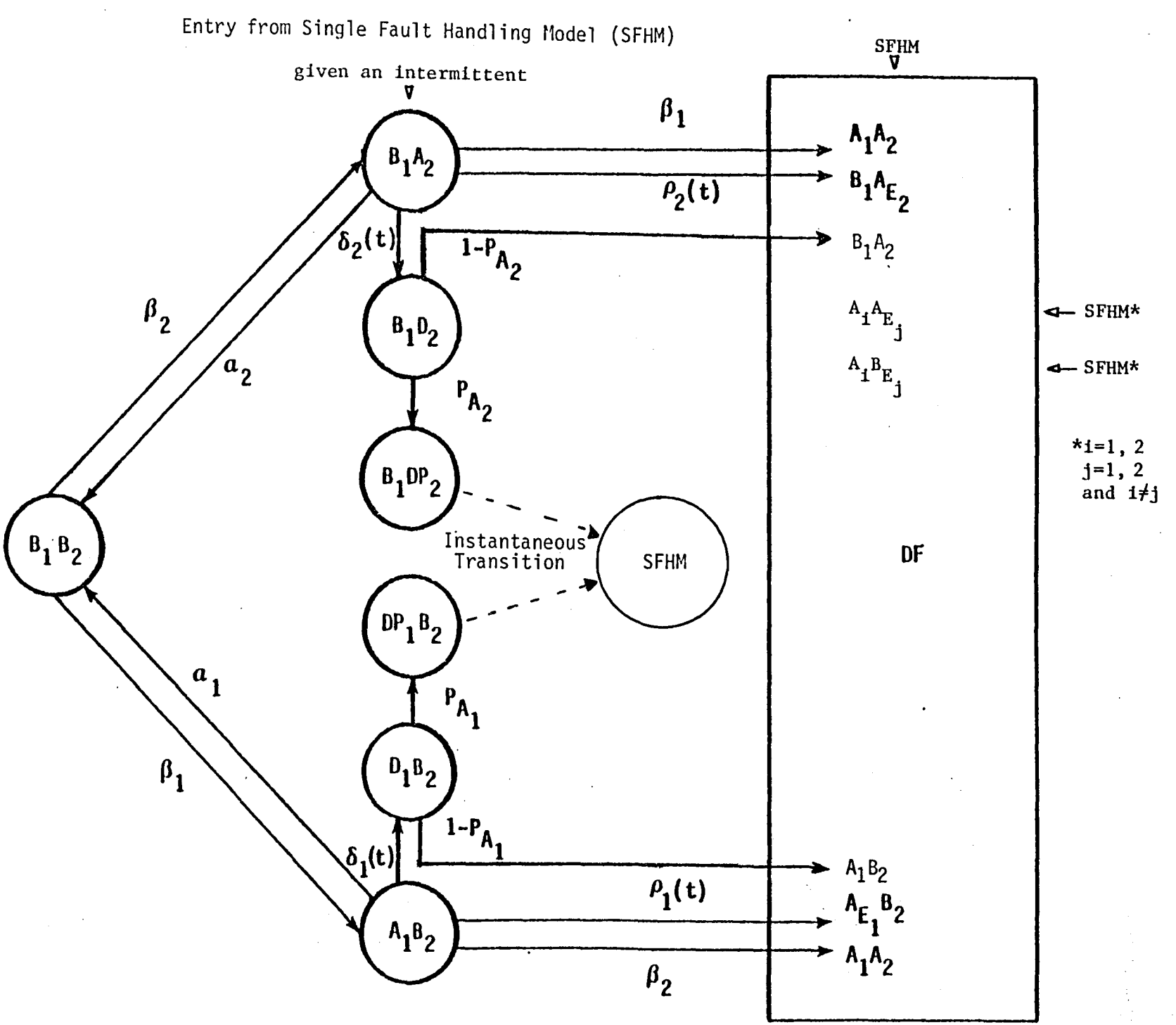


Figure C-1 Illustration of Double Fault-Handling

The manner in which the system handles double faults is depicted by a conceptual illustration of the general double fault handling model given by figure C-1.[1] The parameters of the double-fault model are obtained from the single-fault model automatically requiring no additional user input. In order to keep the model within computational feasibility, CARE III makes the implicit conservative assumption that all critically coupled (user-specified, if any) latent faults will cause system failure. Thus any paired combination of critically coupled faults occupying states A, A_E, or B_E in the single-fault model (see fig. 2-2) will constitute system failure irrespective of whether or not the system still has sufficient operational units left to meet the M out of N stage requirements. Those particular states appear in the large rectangle labeled DF (double failure) as states A_1A_2, A_iA_{Ej}, A_iB_{Ej}, for i=1,2, j=1,2, and $i \neq j$. Entry to these states is by way of the single fault handling model (SFHM).

In addition, CARE III supports an intermittent critically coupled fault model, which is depicted in figure C-1 also by the circled states and state DF. In the event the user defines the single-fault model to represent the system behavior resulting from the occurrence of one or more intermittent faults, the double intermittent fault model will be invoked. On the occurrence of the second fault, the first having been an intermittent fault, state B_1A_2 will be entered. If A_2 also becomes benign, then state B_1B_2 may be entered. Alternately, if while in state A_1B_2 or B_1A_2 the active fault is determined with rate function $\delta_1(t)$ or $\delta_2(t)$, then the fault is determined to be permanent and the system enters state D, the detected state (D_1B_2 or B_1D_2). At this point, CARE III will reconfigure out the faulty detected module (DP_1B_2 or B_1DP_2). The last possible transition is to state DF, the system failure state. That transition can occur if the benign intermittent (B_1 or B_2) becomes active (giving two coexisting critically coupled latent faults), if the active fault begins to propagate errors ($\rho_1(t)$ or $\rho_2(t)$), or if a fault is detected as nonpermanent ($(1-P_{A1})\delta_1(t)$ or $(1-P_{A2})\delta_2(t)$). P_{Ai} is the probability that the ith fault, i=1,2, is detected as permanent. More detailed models could have been implemented (they were in fact seriously considered), but only at the expense of creating greater computational cost with little justification to

---

1. The illustration in figure C-1 is not intended to describe the math model but presents instead a visual concept. The mathematical implementation is partitioned differently (see ref. 5).

back up the need. The present fault/error-handling models represent a compromise between a conservative model with reasonable detail and computational cost.

APPENDIX D

Model Assumptions and Characteristics

D.1 Appendix D.1 summarizes important assumptions and modeling characteristics of the CARE III program that the user may need to know.

GENERAL MODELING CHARACTERISTICS

Time constants (reciprocal of failure rates for $\omega = 1$) of fault occurrences ($\approx 10^{4}$ hours) should be at least 4 orders of magnitude greater than the fault/error transition times ($\approx 10^{-3}$ hour) in the fault/error-handling model. This time separation will insure that mathematical approximations assumed by CARE III are not compromised.

Transition times separated by more than two orders of magnitude in the general fault-handling model (invoked by use of at least one uniform distribution) may cause numerical instability. This restriction does not apply to the homogeneous Markov solution (MARKOV = 1).

FAULT CHARACTERISTICS

Module faults occur independently with Weibull arrival times. The Weibull rate function, $\lambda(t) = \lambda\omega(\lambda t)^{\omega-1}$, allows time dependent rates (t = operational time), thus allowing for much modeling flexibility. The Weibull rate function reduces to the exponential rate function, $\lambda(t) = \lambda$, when ω equals one.

Faults in spare or in-use modules occur according to the same rate functions, i.e., unit dormancy factors.

MODULE DEPLETION FAILURES

Module and stage failures are statistically independent, i.e., a failure in one module or stage cannot cause another module or stage to fail.

A stage fails if fewer than a specified number of modules are operational.

A system fault tree determines which combinations of stage failures define system failure by module depletion.

SYSTEM FAILURES DUE TO IMPERFECT FAULT HANDLING

Single-Fault Failures: An existing latent (undetected) fault creates errors that are not masked/recoverable by the system.

Double-Fault Failures: The interaction of a fault in a module critically paired with an in-use module containing a latent fault causes system failure. In-use modules that are critically paired may be in the same stage or across two stages. Multiple paired faults are possible.

No interactions of more than two critically coupled modules are considered, i.e., no critical triples, quadruples, etc. The case of a quintuple of modules with a majority voter is not directly modeled by CARE III, but can be approximated: All possible pairs within the quintuple are considered critical, and fault-handling unreliabilities are computed only if three or more faults are present (use LC=3).

FAULT-HANDLING MODEL

The same fault-handling model is used for all operational modules, spare or in-use, within a stage. It is independent of global time, i.e., operational time, is independent of the number of operational modules remaining in a stage, and asumes zero probability of incorrectly deleting a module which has not experienced a fault.

Single Fault Handling Model (SFHM): The SFHM models the latency period of a fault and the dynamics of detection, isolation, error generation, etc. It is defined as a semi-Markov process with exponential and/or uniform transition distributions. It defines faults as permanent, intermittent, or transient as a function of parameters α and β. The fault-handling model was designed for fast transients and intermittents, where α and β would be within 2 or 3 orders of magnitude of the other fault-handling parameters, i.e., δ, ρ, etc. Slower transients and intermittents are possible, but the user is cautioned to their use.

Double Fault Handling Model (DFHM): The DFHM models the latency period of two faults where the faults occur in "critical pairs" of modules and one of the faults is intermittent. The DFHM as described here is implemented mathematically in CARE III and is depicted in figure C-1 with states A_iA_{Ej}, A_iB_{Ej} deleted, and entry into state DF occurs only from states drawn to the left of the DF state. The complete illustration depicts the handling of critical-pair faults and is mathematically implemented differently from the DFHM.

SPARING

Parameter matrix NOP defines the number of spare and in-use modules as a function of the number of faulty modules. Only the in-use modules are subject to critical-pair failures.

Fault dynamics are the same for all operational, spare, and in-use modules within a stage, e.g., spare modules are hot and are being dynamically flexed (in-use and spare modules exchange roles); or equivalently, spares are assumed to be executing a self test program, or the spares have some other fault-handling capability.

RECONFIGURATIONS AND FLEXING OF SPARES

Two cases can be considered:

Dynamic flexing: Spare and in-use modules continuously exchange roles (see NOP section 3.2 and appendix B.1). Module flexing is done in order to reduce the number of latent faults by bringing spare modules into voting triads for single-fault detection and reconfiguration.

Reconfiguration: In the event that an in-use module experiences a fault and is detected and reconfigured out of the stage, a hot spare will replace the deleted in-use module, if one is available. If a fault occurs in a spare and is reconfigured, no further action is taken.

Fault analysis of spare modules in CARE III is the same as for in-use modules, i.e., they have the same fault rates (unit dormancy factor) and fault-handling models. In consequence, the probability that a module is in a given fault status (fault free, latent, etc.) is independent of the status, spare or in-use, of the module. This scheme allows for simple evaluation of the probability of lethal critical pairs of modules.

CRITICAL PAIRS

Set of Critical Pairs: The set of critical pairs consists of pairs of modules which when faulty can lead to system failure. This set is defined by a Critical-Pairs Fault Tree. The numbering of modules in this fault tree corresponds
to functional use.

Functional Numbering of Modules: The numbering of modules in the Critical-Pairs Fault Tree corresponds to functions performed in listed order of decreasing importance, but not to physical units (see appendix B.1).

APPENDIX D

Model Assumptions and Characters

D.2 Appendix D.2 explains how subruns are created and how they should be used.

CARE III provides a modeling decomposition technique called a subrun (sub'-run) to significantly reduce the computation of $Q(t|\underline{\ell})$ probabilities associated with user specified critical-pair trees. By using multiple subruns, the user partitions the state space of fault vectors, $\underline{\ell}$, into disjoint sets which consequently reduces the number of elements in the $\underline{\ell}$ vectors for $Q(t|\underline{\ell})$ computations. To properly utilize this feature, the user must insure that stage failure events are not critically coupled across the multiple subruns. If coupling does exist, the failure events all must be contained in the same subrun fault tree.

A subrun can contain up to twenty stages. Another way to say this is that CARE III can only solve operational probabilities, $P(t|\underline{\ell})$, and fault-handling failure probabilities, $Q(t|\underline{\ell})$, where the $\underline{\ell}$ vector for these functions has at most twenty dimensions. (See definitions in section 2.1.) Multiple subruns become necessary when solving CARE III problems with 21 or more stages. The two factors that affect the creation of subruns are the user-defined critical-pair tree(s), if any, and the input parameter NPSBRN.

If one or more critical-pair trees are specified for a given CARE III problem, each critical-pair tree will generate a subrun. The $\underline{\ell}$ vector for each particular subrun will keep track of faults for each of the stages referenced by the original critical-pair tree. For example, suppose the user were modeling a problem with eight stages, and the user defines two critical-pair trees. The first critical-pair tree references stages one thru three in its STAGE ID AND MODULE RANGE lines, and the second critical-pair tree references stages four thru eight. This problem would generate two subruns. The first subrun's $P(t|\underline{\ell})$ and $Q(t|\underline{\ell})$ arrays would be a function of an $\underline{\ell}$ vector having three dimensions. The first element of this vector would keep track of faults for stage one. Similarly, the second element of this vector would keep track of faults for stage two, and the third element would keep track of faults for stage three. The second subrun's $P(t|\underline{\ell})$ and $Q(t|\underline{\ell})$ arrays would be a function of an $\underline{\ell}$ vector having five dimensions. The first element of this vector would

keep track of faults for stage four, the second element would keep track of faults for stage five, and so on, with the last element of this subrun's $\underline{\ell}$ vector keeping tracking of faults in stage eight.

When solving for the total fault-handling unreliability QSUM array printed in the summary information listing, the CARE III program assumes that stages grouped by subruns are in separate branches of the system tree such that the branches are 'ORed' together. (A branch to the system tree is defined to be a portion of the tree which serves as an input to the top event of the tree.) If this assumption is violated, the total fault-handling unreliability as given by QSUM may not be conservative. When the above assumption is violated, some individual $Q(t|\underline{\ell})$ values, possibly important to the QSUM total, will not be computed. To obtain details of the mathematics of the above assumption, the reader can find information in section 4.0 of reference 6 and in section 3.5.4 of reference 5.

If all of the stages for a given CARE III problem are not contained entirely in one or more critical-pair trees or if the CARE III program has no critical-pair tree, the CARE III program will automatically create subruns from the stages not referenced by a critical-pair tree. The algorithm that CARE III uses to do this can be controlled somewhat by using the input parameter NPSBRN. If no critical-pair tree exists, then NPSBRN will partition the stages of the problem being solved into subruns with 'NPSBRN' stages to a subrun, e.g., a CARE III problem consisting of a twenty stage system with NPSBRN = 5 will consist of four subruns with five stages per subrun. For this example subrun, one's calculations would be based on an $\underline{\ell}$ vector keeping track of stages 1-5, subrun two would similarly keep track of stage 6-10, subrun three would keep track of stages 11-15, and subrun four would keep track of stages 16-20. If a single critical-pair tree exists such that not every stage is referenced by the tree, the excluded stages will form subruns using the NPSBRN parameter. Let's now suppose one is working a CARE III problem containing twenty stages where the first 14 are referenced in the STAGE ID AND MODULE RANGE section of a single critical-pair tree and that stages 15 thru 20 are free of critically-paired faults. If NPSBRN = 5 for this example, then the CARE III program will divide this problem into three subruns. Subrun one will consist of stages 1-14, subrun two will consist of stages 15-19, and subrun three will only consist and use stage number twenty. Thus the algorithm used by the CARE III program to

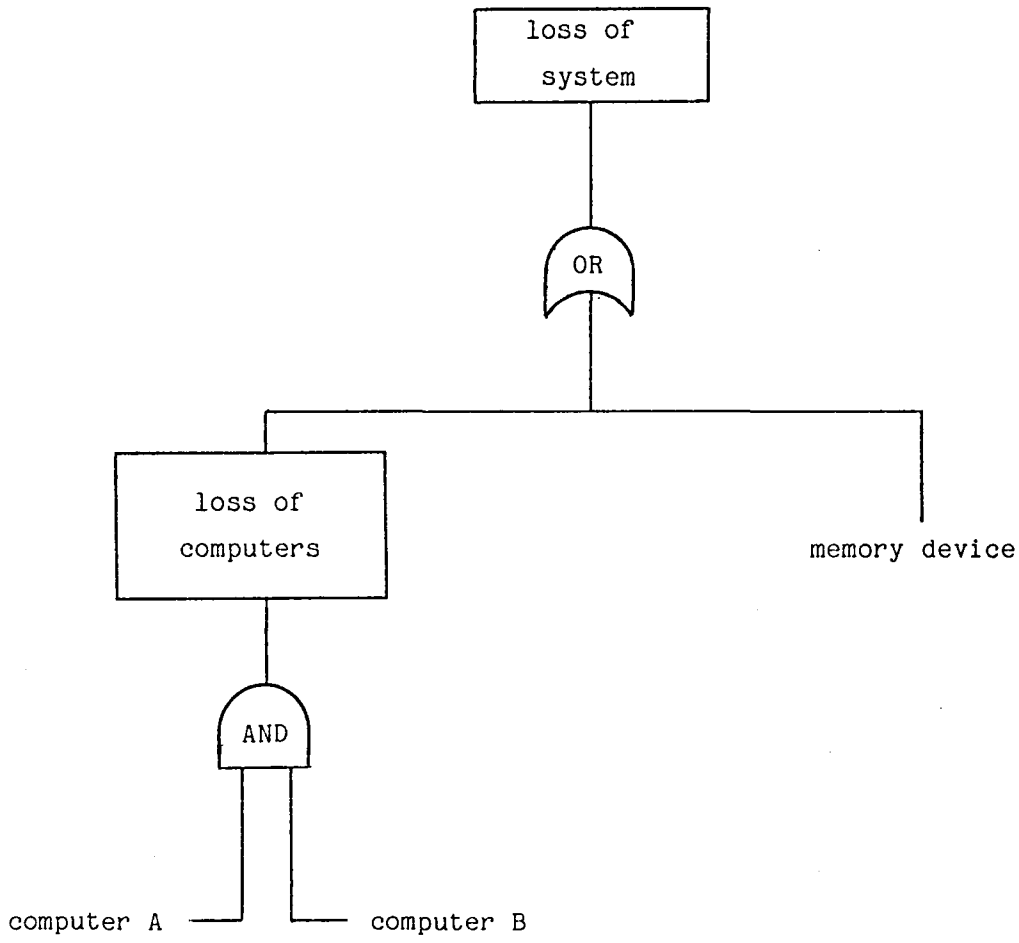
create subrun works as follows. Every critical-pair tree provided by the user creates a subrun consisting of the stages referenced by its STAGE ID AND MODULE RANGE section. After this, stages not referenced by a critical-pair tree are put into subruns that are different from those created by the critical-pair trees. In doing this, however, stages referenced by a subrun must be in sequential order. Stages preceded and followed by the stages used in a critical-pair tree could not be grouped together, for instance. Finally, for the groups of sequentially numbered stages that are not referenced by a critical-pair tree, these stages are further divided into groups so that subruns generated from these groups will be multiples of the NPSBRN value plus remainder, e.g., with NPSBRN = 5 and with a gap of 12 sequentially numbered stages, three subruns will be created from this gap. The first two would consist of five stages each, and the third would consist of two stages.

The advantage of using NPSBRN to create subruns is that the CARE III program will execute faster when solving $Q(t|\underline{\ell})$ and $P(t|\underline{\ell})$ probabilities with $\underline{\ell}$ having as few dimensions as possible. However, by creating many subruns there may be more $Q(t|\underline{\ell})$ and $P(t|\underline{\ell})$ probabilities to solve for. As a trade off, it is usually best to put two to four stages in a subrun when possible.

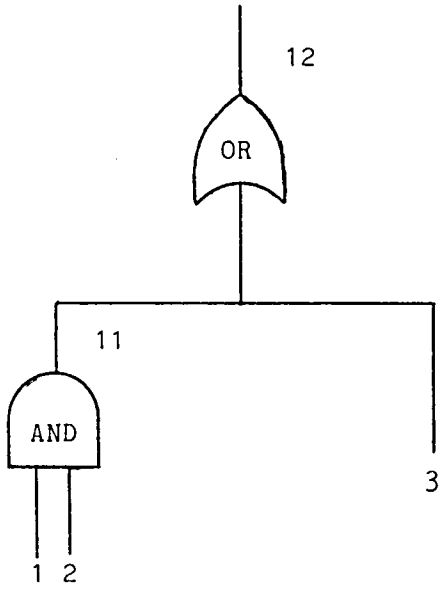
The reader should be aware of the following caveats at this time. The depletion of hardware probability, P*SUM, printed in the summary information listing is computed separately from any $Q(t|\underline{\ell})$, $P(t|\underline{\ell})$, and QSUM calculations. The P*SUM answer will also remain the same regardless of how the problem of interest is divided into subruns. Even though the user may only want to use CARE III to calculate P*SUM, the CARE III program will calculate $P(t|\underline{\ell})$ arrays for all subruns even when no critical-pair trees exist and parameter IRLPCD = 1. Unfortunately, the present version of the CARE III program does not allow the $P(t|\underline{\ell})$ calculations pertaining to subruns to be disabled when only P*SUM calculations are desired. If subruns are not partitioned appropriately through the use of critical-pair trees and the parameter NPSBRN, the $P(t|\underline{\ell})$ calculations may take a long time leading to long CARE III execution times. The other point to note is that for any CARE III problem involving 21 or more stages which has a system tree with the tree's top gate being an 'AND' or 'M of N' gate, the total fault-handling unreliability, QSUM, cannot in general be assumed to be conservative. Furthermore, the second release version of the CARE III program does not warn the user if he violated the assumption

about subruns containing stages which are located in system tree branches which are not 'ORed' together when there are critical-pair failure calculations to be done. Examples are now illustrated to aid in identifying if this assumption has been violated.

Consider a system made of three stages. Two of the stages are computer stages with the third stage being a memory storage device. Let's say computer stage A has four modules configured so that computer stage A starts as a triad with one spare and that computer stage B is configured similarly. Let's also assume that the memory device stage consists of one module and is shared between the two computer stages. We will consider the system as being failed if both computer stages fail or if the memory storage device fails. We will also consider critical-pair failures in both working computer triads, e.g., system failure occurs if a computer triad incurs two faults in succession such that the second fault occurs before the first fault is dealt with. With the above information, we can draw the system failure tree which looks like this -



If we associate computer stage A to be stage number one, and computer stage B to be stage number two, and the memory storage device to be stage number three, then the system tree could be numbered like so -



If we reference all three stages in one critical-pair tree, the bottom portion of the input file for this problem would look like -

```
SYSTEM FAULT TREE
 1  3 11 12
11  A  1  2
12  0  3 11
CRITICAL PAIR TREE
 1  9 21 23
 1  1  4
 2  5  8
 3  9  9
21  2  1  2  3
22  2  5  6  7
23  0 21 22
```

With this type of input, our CARE III problem will be done as one subrun. We notice, however, that stage number three's module is not referenced in the logic block of the critical-pair tree. We also note that stage three is connected to the top event of the system tree through an OR gate. This structure satisfies the assumption of being in a separate branch of the system fault tree which is 'ORed' together with one or more other branches of the tree. The above input can be simplified to -

```
SYSTEM FAULT TREE
 1   3 11 12
11   A  1  2
12   0  3 11
CRITICAL PAIR TREE
 1   8 21 23
 1   1  4
 2   5  8
21   2  1  2  3
22   2  5  6  7
23   0 21 22
```

This input will now produce two subruns. Subrun one will have $P(t|\underline{\ell})$ and $Q(t|\underline{\ell})$ values pertaining to stages one and two, and subrun two will have $P(t|\underline{\ell})$ and $Q(t|\underline{\ell})$ values pertaining to stage three.

In the above example, there are no critical-pair failures involving both computer stage A and computer stage B. The critical-pair failures are within stages and not across stages. Because of this, the user may think of breaking the above problem into three subruns by using the input -

```
SYSTEM TREE
 1   3 11 12
11   A  1  2
12   0  3 11
CRITICAL PAIR TREE ONE
 1   4 21 21
 1   1  4
21   2  1  2  3
CRITICAL PAIR TREE TWO
 1   4 21 21
 2   1  4
21   2  1  2  3
```

This input, however, would violate the assumption that stages one and two are in separate branches of the system fault tree and much less that the branches are 'ORed' together by the top event of the tree. If the user were to use this type of input to execute the CARE III program, the program may compute QSUM values in the QSUM array that may not be conservative.

Let's now analyze the $Q(t|\underline{\ell})$ arrays generated by CARE III for the different variations of the example discussed. For this comparison, let's assume both computer stages use a permanent fault-handling model with the parameter C=1.0.

If the above example is done in one subrun, up to 17 nonzero valued $Q(t|\underline{\ell})$ arrays would be computed. They would be Q(0,2,0), Q(2,0,0), Q(0,3,0), Q(1,2,0), Q(2,1,0), Q(3,0,0), Q(0,4,0), Q(1,3,0), Q(2,2,0), Q(3,1,0), Q(4,0,0), Q(1,4,0), Q(2,3,0), Q(3,2,0), Q(4,1,0), Q(2,4,0), and Q(4,2,0). Since this problem only deals with permanent faults, these arrays, of the form Q(x,y,z), give the probability of failure due to fault-handling with x modules of stage one having failed, y modules of stage two having failed, and z modules of stage three having failed. The above 17 $Q(t|\underline{\ell})$ arrays would be added to form the QSUM array printed in the summary information listing.

For the case in which the above problem was done in two subruns, the first subrun would generate a maximum of 17 nonzero valued $Q(t|\underline{\ell})$ arrays, and the second subrun would not generate any nonzero valued $Q(t|\underline{\ell})$ arrays. For the first subrun, the $Q(t|\underline{\ell})$ arrays of interest would be Q(0,2), Q(2,0), Q(0,3), Q(1,2), Q(2,1), Q(3,0), Q(0,4), Q(1,3), Q(2,2), Q(3,1), Q(4,0), Q(1,4), Q(2,3), Q(3,2), Q(4,1), Q(2,4), and Q(4,2).

If the example problem had been done using three subruns, the only nonzero valued $Q(t|\underline{\ell})$ arrays would be generated in subrun one and in subrun two. For subrun one, these would be Q(2), Q(3), and Q(4). For subrun two, these would also be Q(2), Q(3), and Q(4). Notice that only six nonzero valued $Q(t|\underline{\ell})$ arrays are calculated. For this case, not all combinations of fault-handling failures have been examined so the QSUM array values may not be conservative.

APPENDIX E

NAMELIST and List-Directed Syntax

E.1 Syntax Rules for NAMELIST Input

The CARE III program permits the use of NAMELIST formatted input for input paragraphs FLTTYP, STAGES, FLTCAT, and RNTIME. NAMELIST statements permit the input of groups of variables and arrays with identifying names. NAMELIST input is an extension of ANSI standard Fortran 77. The following rules should be observed when using the VAX-11 Fortran or CDC Fortran V compilers.

1) A NAMELIST paragraph begins with a $ in column two, immediately followed by the NAMELIST paragraph identifier. At least one space, and spaces only, must separate the NAMELIST paragraph identifier and the first identifying name of the first variable or array used in the NAMELIST paragraph. The first variable or array name may, however, start the next line following the NAMELIST paragraph identifier.

2) Data items in a NAMELIST paragraph are separated by commas and may be in the forms:
   a) variable = constant
   b) array name = constant, ..., constant
   c) array name(subscript) = constant, ..., constant

   Column one must always be left blank when formatting NAMELIST data.[1] Forgetting the constant behind a variable or array name will create a fatal execution time error.

3) Any data item in a NAMELIST paragraph which is not assigned by the user will be given a default value as shown by the table in section 3.7.

4) The order in which variables or array names are listed in a NAMELIST paragraph is not important.

---

1. Column one of the input file may be used when inputting the CARE III system fault tree or critical-pair trees.

6) Many lines can be used to input data in a NAMELIST paragraph. Column one of each new line must be kept blank. All constants in a NAMELIST paragraph must be followed by a comma, except for the last constant in the NAMELIST paragraph. The last constant is followed by a $ or $END, or by both a comma and $.

7) The CARE III program reads integer, real, and logical data types. See the host machine's Fortran Reference Manual for acceptable data type representations.

8) Constants can be preceded by an asterisk repetition factor.

```
RLM = 7*1.0E-3,
```

is equivalent to

```
RLM(1,1) = 0.001,  RLM(2,1) = 0.001,  RLM(3,1) = 0.001,
RLM(4,1) = 0.001,  RLM(5,1) = 0.001,  RLM(1,2) = 0.001,
RLM(2,2) = 0.001,
```

Also -

```
ALP(2) = 2*750.0, 2*1600.0,
```

is equivalent to

```
ALP(2) = 750.0, ALP(3) = 750.0, ALP(4) = 1600.0, ALP(5) = 1600.0,
```

9) Blanks must not appear -

   a) between $ and NAMELIST paragraph name
   b) within a NAMELIST paragraph name
   c) within array names or variable names
   d) within constants
   e) on either side of the asterisk repetition symbol

10) Spaces may appear before or after commas, before or after equal signs, and before or after parentheses.

E.2 Syntax Rules for List-directed Input

The user may select list-directed formatting for input paragraphs FLTTYP, STAGES, FLTCAT, and RNTIME. List-directed format is not as easy to use as the NAMELIST format mostly because the order in which parameters are listed for list-directed formatting is important. For a more detailed listing of the general rules of list-directed formatting, see the host machine's Fortran Reference Manual.

GENERAL RULES

1) List-directed data is read beginning in column one starting with the first line of the input file.

2) A slash divides input belonging to different input paragraphs. After the slash is encountered, data for a new input paragraph must start on a new line.

3) Data for the variables and arrays of the input paragraphs must be entered in the order shown by the table given in section 3.7.

4) A comma must come after every input parameter entered. An input may be a constant, a null value indicated by a space, a repetition of constants in the form r*c, and a repetition of null values in the form r* , where r is an integer specifying the number of repetitions, and c is the constant to be replicated.

5) If not all parameters for an input paragraph have been listed before a slash is encountered in the input stream, any parameters not accounted for will stay at their default value as given by the table shown in section 3.7.

6) Spaces may not appear -
   a) within constants
   b) on either side of the asterisk repetition symbol

7) The CARE III program reads integer, real, and logical data types for its various input parameters. See the host machine's Fortran Reference Manual for acceptable data type representations.

8) To leave a parameter at its default value, a null value must be entered for it. If the parameter would have occupied the first position of the input paragraph, an initial comma is inserted for the parameter, otherwise two consecutive commas with no intervening constants enters the null value. Null values at the end of the input paragraph can be entered depending upon the location of the input paragraph's terminating slash (see rule 5). One or more spaces is allowed on either side of a comma.

SPECIFIC RULES

The number of input values required by some input arrays when using list-directed formatting will vary each time the user creates a new CARE III problem. These arrays and the number of data required by CARE III using list-directed formatting is now discussed.

For input paragraph FLTTYP, the CARE III program will read 'NFTYPS' elements for the arrays ALP, BET, DEL, RHO, EPS, IDELF, IRHOF, IEPSF, PA, PB, and C. As an example, if NFTYPS = 3, then the CARE III program expects three input constants for each of the above arrays in the order they are listed.

For the input paragraph STAGES, the input NSTAGES will determine how many inputs will be needed for arrays N, M, NOP, and LC. 'NSTGES' inputs will be needed for the arrays N and M. The number of inputs for the NOP array will be five times 'NSTGES', e.g., five values of NOP are needed for each stage of a CARE III problem. Finally, 'NSTGES' inputs will be needed for array LC.

For input paragraph FLTCAT, the CARE III program expects 'NSTGES' inputs for the first array NFCATS. The quantity of fault-handling models that the ith stage has as given by array NFCATS, will determine the number of inputs to be expected for the two-dimensional arrays JTYP, OMG, and RLM. The sum of array NFCATS, over elements NFCATS(1) to NFCATS(NSTGES), will be the number of inputs required for these arrays.

The input paragraph RNTIME has no arrays; however, the order in which input variables are listed is still very important.

Below is shown the Fortran code for the CAREIN module that uses list-directed formatting. It is provided for the user so he can see the exact way in which the input paragraphs FLTTYP, STAGES, FLTCAT, and RNTIME are read using list-directed formatting.

```
      READ(7,*,END=10)       NFTYPS
     .,  (  ALP(ITYP),ITYP=1,NFTYPS)
     .,  (  BET(ITYP),ITYP=1,NFTYPS)
     .,  (  DEL(ITYP),ITYP=1,NFTYPS)
     .,  (  RHO(ITYP),ITYP=1,NFTYPS)
     .,  (  EPS(ITYP),ITYP=1,NFTYPS)
     .,  (IDELF(ITYP),ITYP=1,NFTYPS)
     .,  (IRHOF(ITYP),ITYP=1,NFTYPS)
     .,  (IEPSF(ITYP),ITYP=1,NFTYPS)
     .,  (   PA(ITYP),ITYP=1,NFTYPS)
     .,  (   PB(ITYP),ITYP=1,NFTYPS)
     .,  (    C(ITYP),ITYP=1,NFTYPS)
     .,  DBLDF ,TRUNC ,CVPRNT,CVPLOT,IAXSCV,MARKOV,LGTMST
      READ(7,*,END=10)                  NSTGES
     .,  (   N(   ISTG),         ISTG=1,NSTGES)
     .,  (   M(   ISTG),         ISTG=1,NSTGES)
     .,  ((NOP(IQ,ISTG),IQ=1,5),ISTG=1,NSTGES)
     .,  (  LC(   ISTG),         ISTG=1,NSTGES)
     .,  IRLPCD,RLPLOT,IAXSRL
      READ(7,*,END=10)           (NFCATS(ISTG) ,ISTG=1,NSTGES)
     .,  ((JTYP(ICAT,ISTG),ICAT=1,NFCATS(ISTG)),ISTG=1,NSTGES)
     .,  (( OMG(ICAT,ISTG),ICAT=1,NFCATS(ISTG)),ISTG=1,NSTGES)
     .,  (( RLM(ICAT,ISTG),ICAT=1,NFCATS(ISTG)),ISTG=1,NSTGES)
      READ(7,*,END=10)
     .   FT    ,NSTEPS,ITBASE,SYSFLG,CPLFLG,KWT    ,PSTRNC
     .,  CINDBG,QPTRNC,IVSN  ,NPSBRN,CKDATA
```

APPENDIX F

Control Parameters

TRUNC - fault-handling function's truncation value for the general fault-handling model solution. Recall that the general fault-handling model is solved when the parameter MARKOV equals two. The TRUNC parameter limits the number of points calculated for the single and double fault-handling functions. When these functions become less than TRUNC, no more of their points are calculated.

The value of TRUNC usually has little affect on the execution time of a CARE III job. Using smaller values of TRUNC will in most cases provide a more accurate fault-handling unreliability value (i.e., more accurate $Q(t|\underline{\ell})$ values used to get QSUM). The amount of accuracy gained, however, is usually small.

LC - parameter that can inhibit $Q(t|\underline{\ell})$ contributions to fault-handling unreliability, QSUM, when critical-pair failures exist. LC is specified for each stage of the system and the default for all LC values is zero. LC may allow for the approximate modeling of some reconfigurable (e.g. pentaplex) systems.

As shown by the definition of $Q(t|\underline{\ell})$ in section 2.1, a given $Q(t|\underline{\ell})$ probability is conditional on the value of $\underline{\ell}$. The LC parameter inhibits a critical-pair failure computation of a particular $Q(t|\underline{\ell})$ probability when the vector $\underline{\ell}$ for the $Q(t|\underline{\ell})$ computation has an element less than LC. For non-transient faults, this element of $\underline{\ell}$ represents the number of faults incurred by stage x. For transient faults, this element of $\underline{\ell}$ represents the number of modules reconfigured from the system.

The parameter LC will not affect any single-fault failure calculations when the parameter C, used to describe the fault-handling model, is less than one. Unreliability due to single-fault failures is added to the QSUM array regardless of the parameter LC.

PSTRNC- parameter used to limit the number of $Q(t|\underline{\ell})$ values used in computing fault-handling unreliability, QSUM. If the maximum value of $P^*(t|\underline{\ell})$, for t over all mission time, is less than 'PSTRNC', the corresponding $Q(t|\underline{\ell})$ value will not be used in computing fault-handling unreliability, QSUM.

The parameter FT which specified the operational time will affect how PSTRNC works because the module depletion probability, $P^*(t|\underline{\ell})$, for a given fault vector, $\underline{\ell}$, changes when the mission time, FT, changes. For a given value of PSTRNC, using a large value for FT will usually cause the CARE III program to calculate fewer $P(t|\underline{\ell})$ and $Q(t|\underline{\ell})$ values than a small value for FT will.

Any condition which causes fewer $P(t|\underline{\ell})$ and $Q(t|\underline{\ell})$ probabilities to be calculated will decrease the execution time of the CARE III program. If too large of a value of PSTRNC is used for a given problem, however, it is possible that some of the more significant $Q(t|\underline{\ell})$ values or possibly no $Q(t|\underline{\ell})$ values at all will be calculated. This value of PSTRNC in turn would make the value of QSUM on the output summary information page smaller than it should be.

QPTRNC- parameter used to limit the number of $Q(t|\underline{\ell})$ values used in computing fault-handling unreliability. If a problem being worked has some of its total unreliability due to fault handling, the CARE III program will print the most dominate $Q(t|\underline{\ell})$ values making up this unreliability. After the most dominate (largest value) $Q(t|\underline{\ell})$ values are computed, any $Q(t|\underline{\ell})$ values which are a factor of 'QPTRNC' smaller than these values will not be calculated.

If QPTRNC is set large, say a value of one or greater, only the most dominate $Q(t|\underline{\ell})$ values will be calculated and the final QSUM answer may not be conservative due to the many lesser individual $Q(t|\underline{\ell})$ terms which are dropped from the sum. Using QPTRNC values less than the default value of 1.0E-2, may cause the execution time of large problems to be intolerably long while providing very little improved accuracy.

where the input data have been thoroughly debugged, and only a few minor changes are made to the input file for subsequent runs. This capability is also useful when CARE III is used with a user-friendly preprocessor that does the data checking in an interactive manner.

NPSBRN- parameter that is used to determine the number of subruns and consequently, the number of noncritically-paired stages that are to be grouped together in a subrun. The default value is 20, but faster execution will usually result if a value of three or four is used (see appendix D for more detail).

APPENDIX G

Example Problems

G.1 Example System Trees

A number of system fault trees have been sketched for some typical aircraft flight control system architectures. The figure numbers (5 - 11) correspond to those given in reference 18. Further details on the system architectures can be found in reference 18.

Two types of notation are used in the examples: expanded and condensed. Expanded notation is usually used if devices have different failure rates and/or there are failure dependencies (see fig.'s 7, 8, and 11). Expanded notation can also be used to show detail, e.g., fig.'s 5, 6, 9, 10. Typically, in the latter case, condensed notation is used.

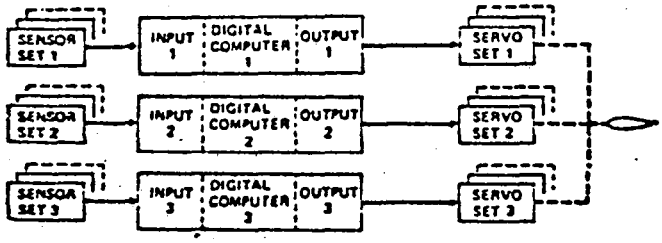
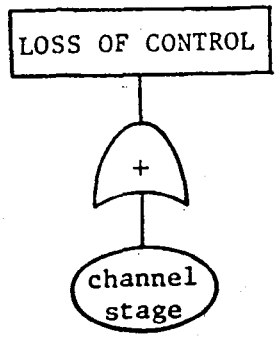
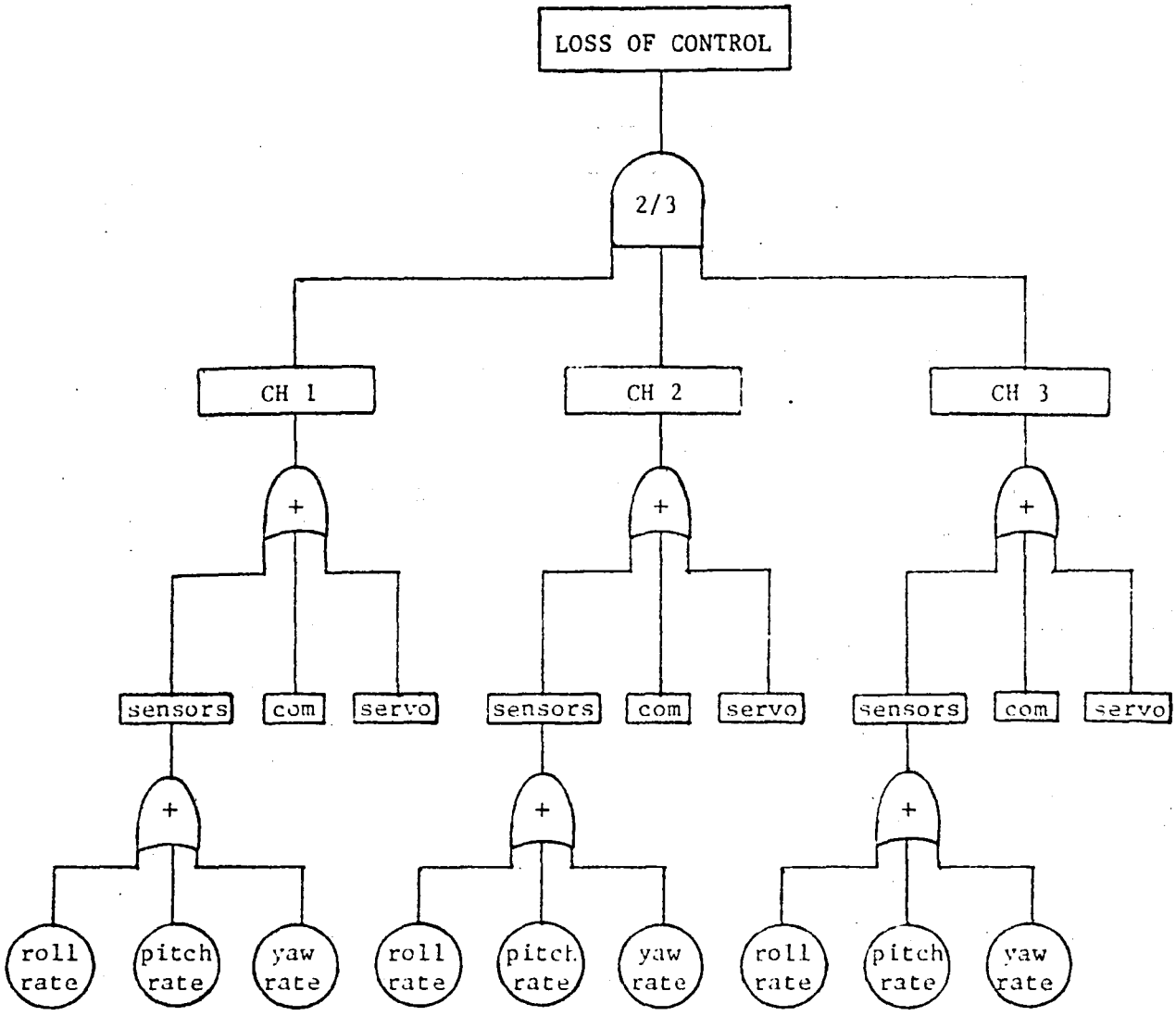


Figure 5. Triplex Voted System–Servo Force Voting Only



N = 3 M = 2

CARE III CONDENSED NOTATION



CARE III EXPANDED NOTATION

FIGURE 5

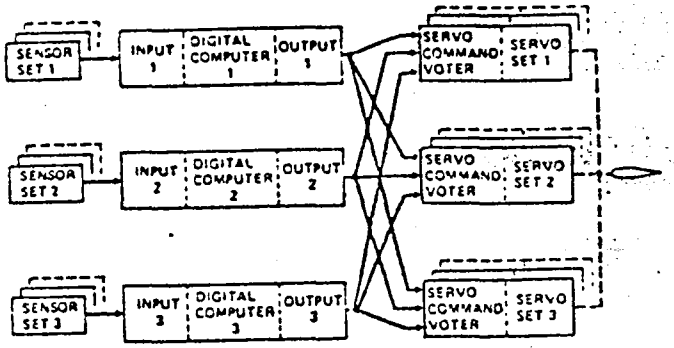
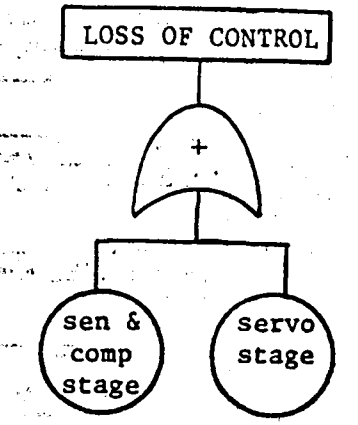


Figure 6. Triplex Voted System—Servo Force Voting and Servo Command Voting



N=3, M=2 N=3, M=2

CONDENSED NOTATION

EXPANDED NOTATION

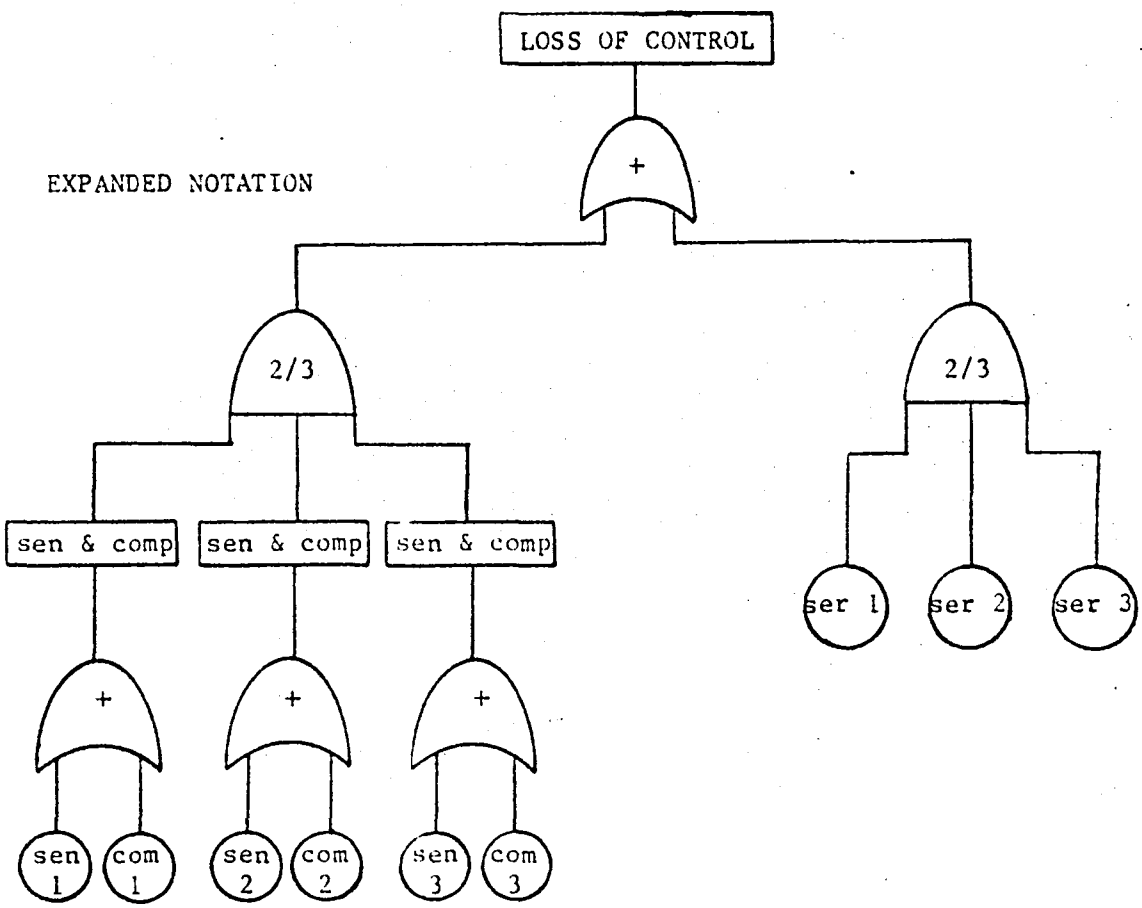


FIGURE 6

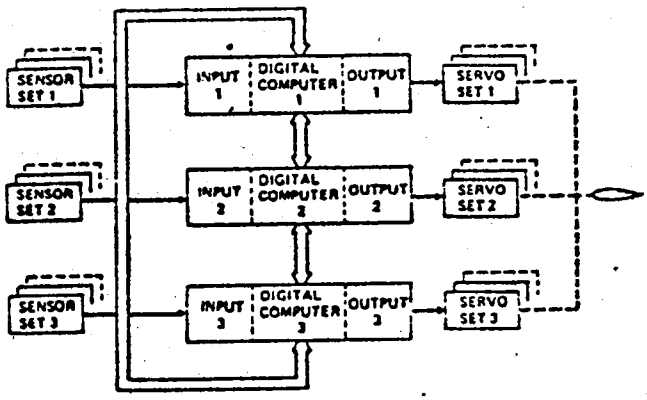


Figure 7. Triplex Voted System—Servo Force Voting and Sensor Signal Voting

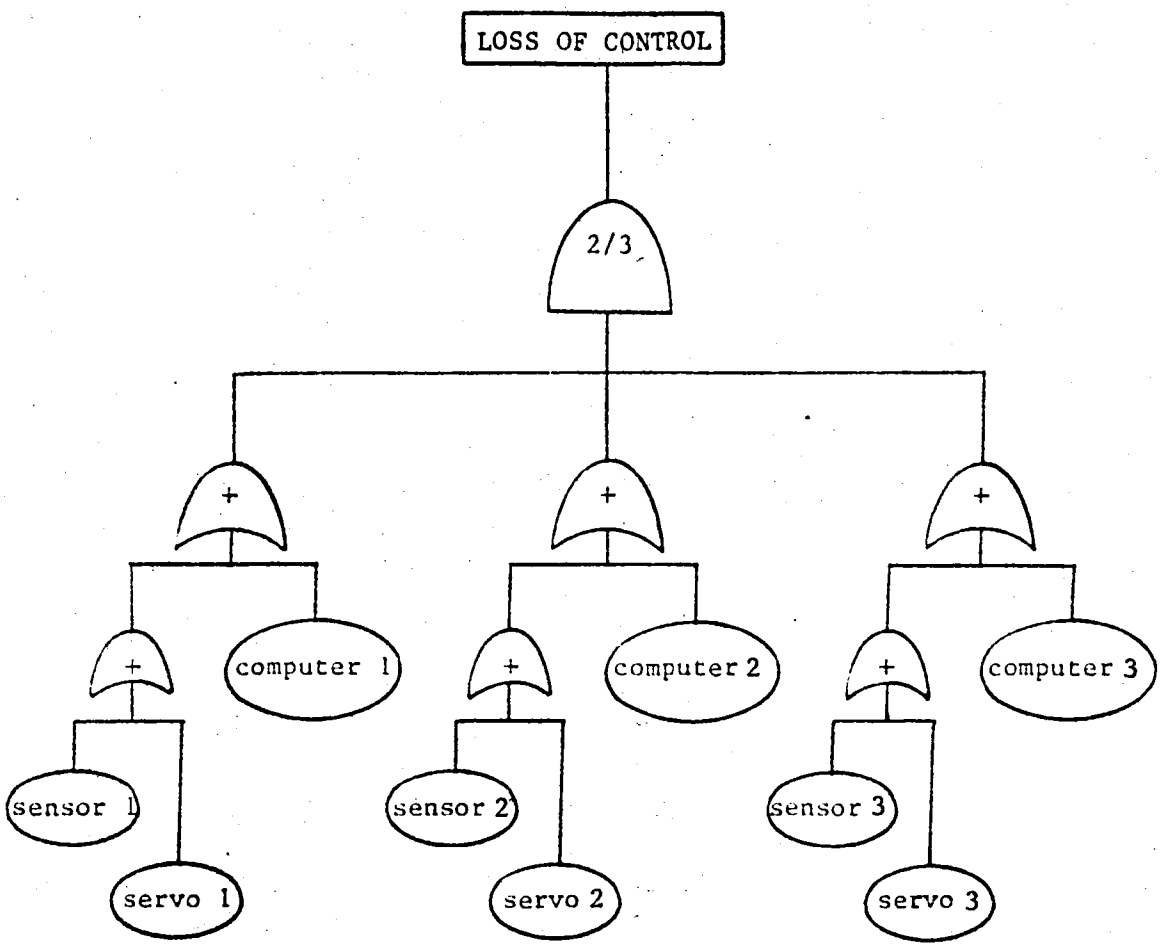


FIGURE 7

NO CONDENSED EQUIVALENT

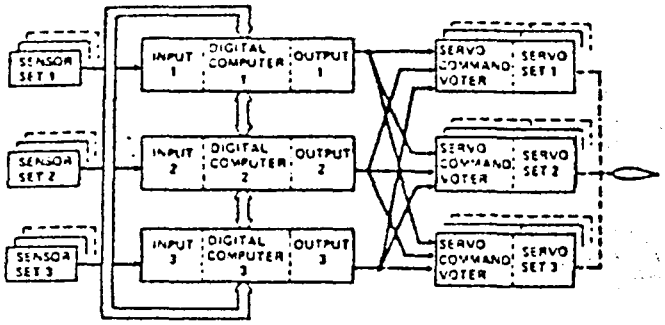
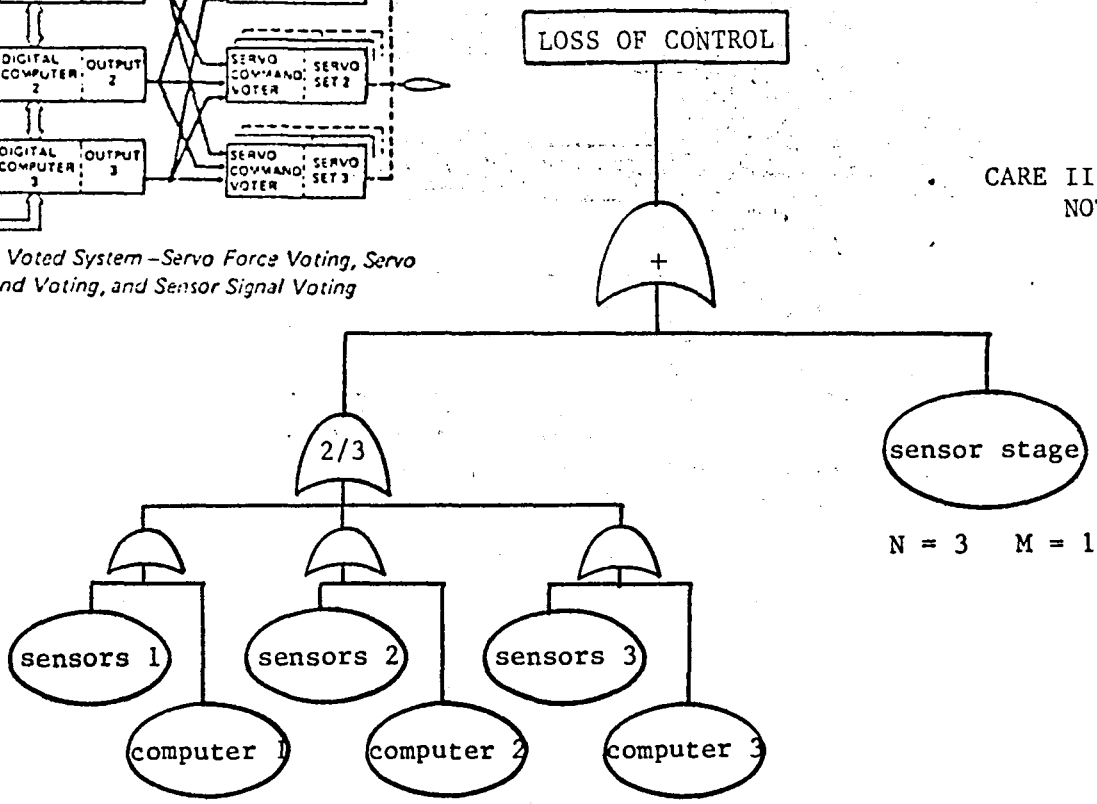
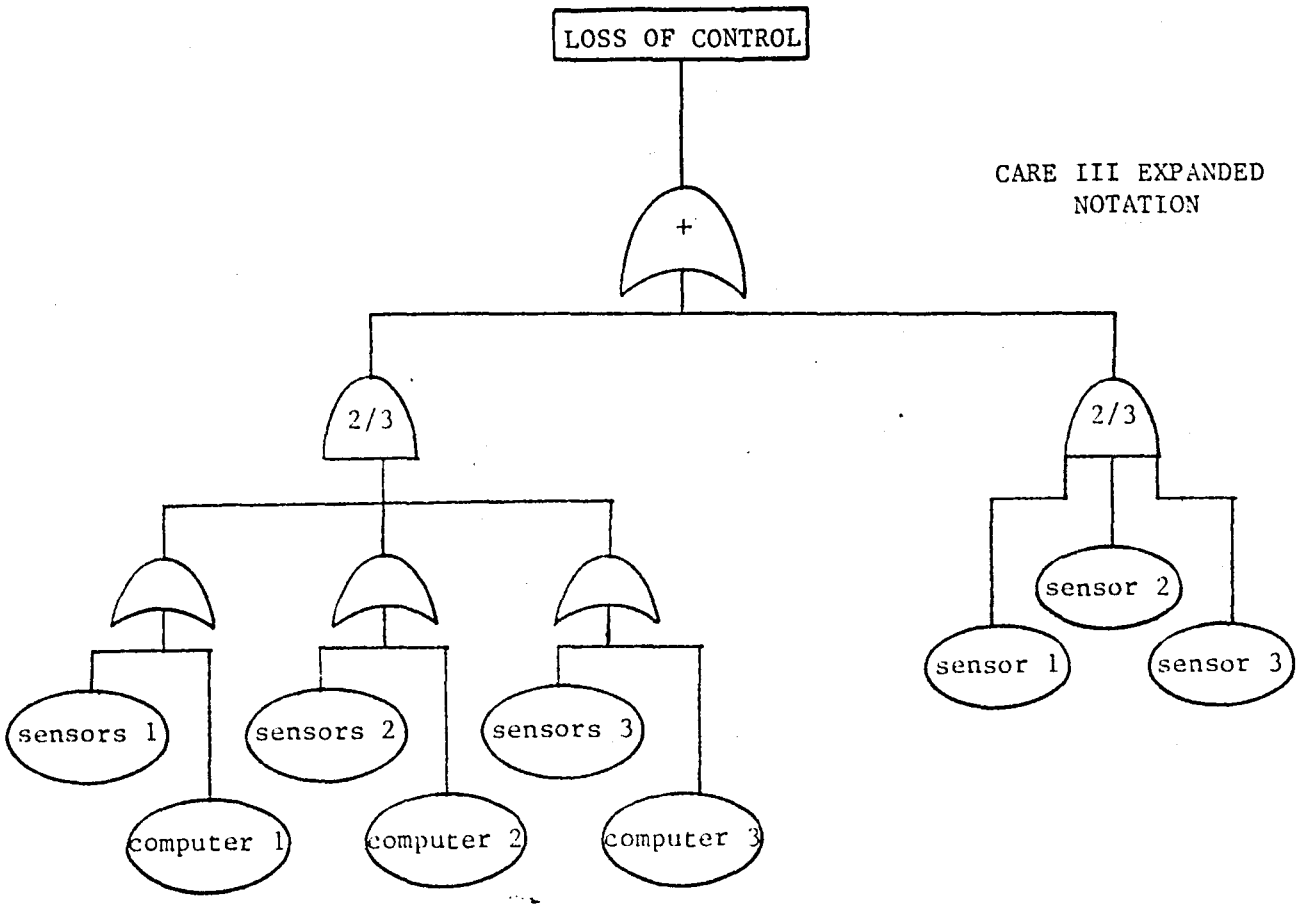


Figure 8. Triplex Voted System – Servo Force Voting, Servo Command Voting, and Sensor Signal Voting



CARE III CONDENSED NOTATION

N = 3 M = 1



CARE III EXPANDED NOTATION

FIGURE 8

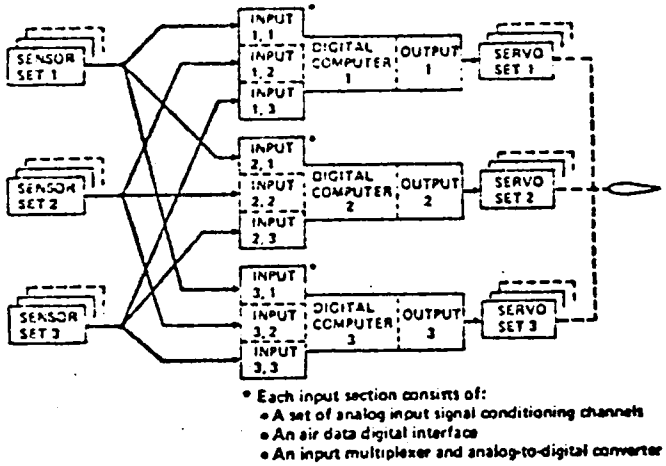


Figure 9. Triplex Voted System–Servo Force Voting and Cross-Strapped Sensor Signal Voting

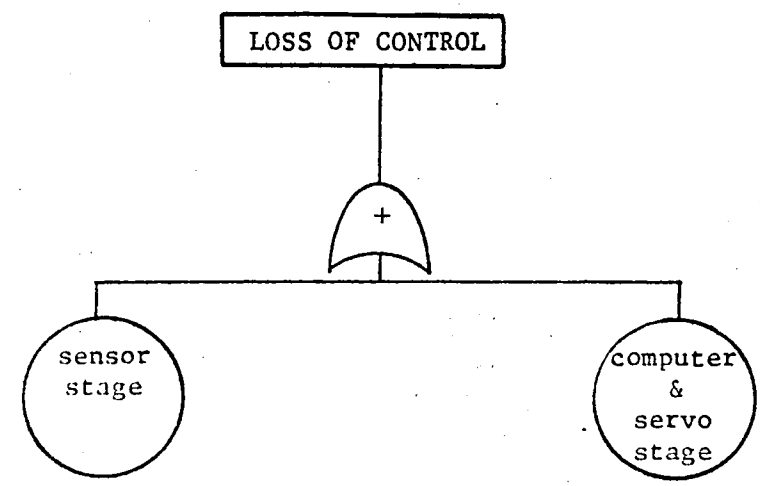


FIGURE 9

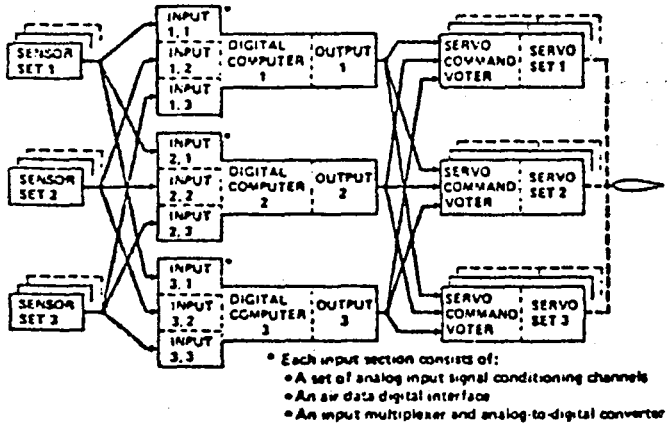
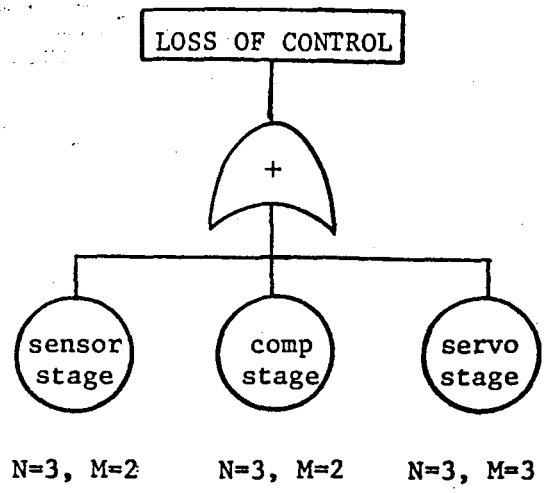


Figure 10. Triplex Voted System–Servo Force Voting, Servo Command Voting, and Cross-Strapped Sensor Signal Voting



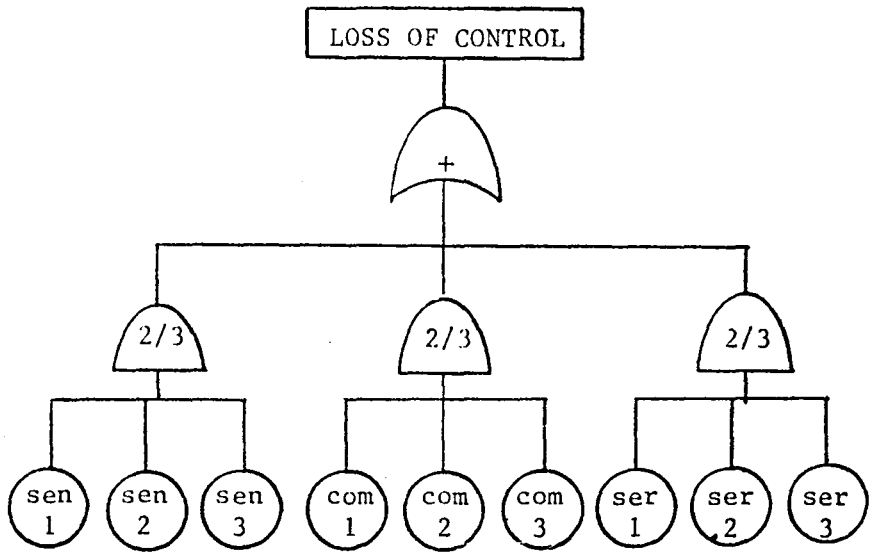


FIGURE 10

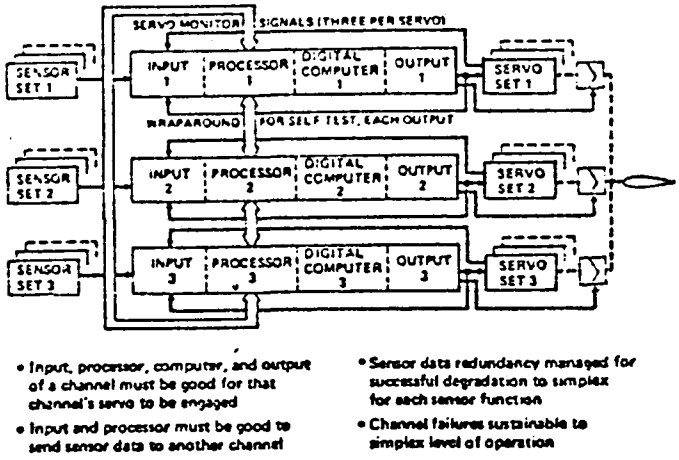


Figure 11. Triplex ARCS Concept

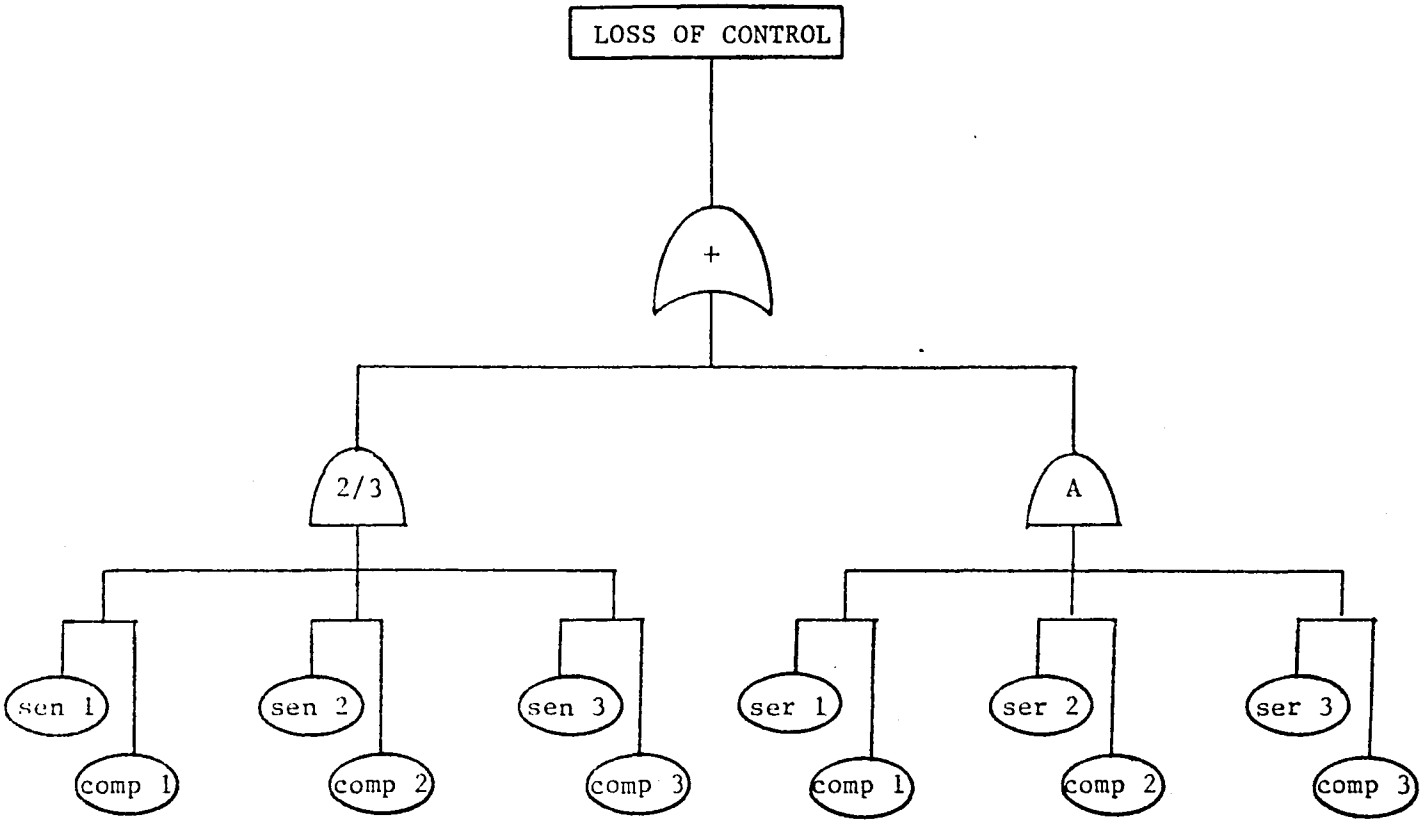


FIGURE 11

APPENDIX G

Example Problems

G.2 Complete Examples

Two example problems that are presented in the "Hands-On Demonstration and Tutorial" report (ref. 3) are included in abbreviated form in this appendix. For each problem, the input file is given together with a sample (incomplete) of the output data. The output listings shown, however, are not those shown in reference 3. The output listings shown are those generated by the new release version 2 code. Since release 2 uses an improved variable time step integrator, the data listed in this guide will usually not correspond to those in reference 3, since the time steps will usually be different. By selecting the original linear time step integrator (setting LGTMST = .FALSE.), the time steps can be made to correspond. Also, the accuracy of the CARE III computations has increased, so the user will see small differences in the output when comparing answers to those of past versions of CARE III.

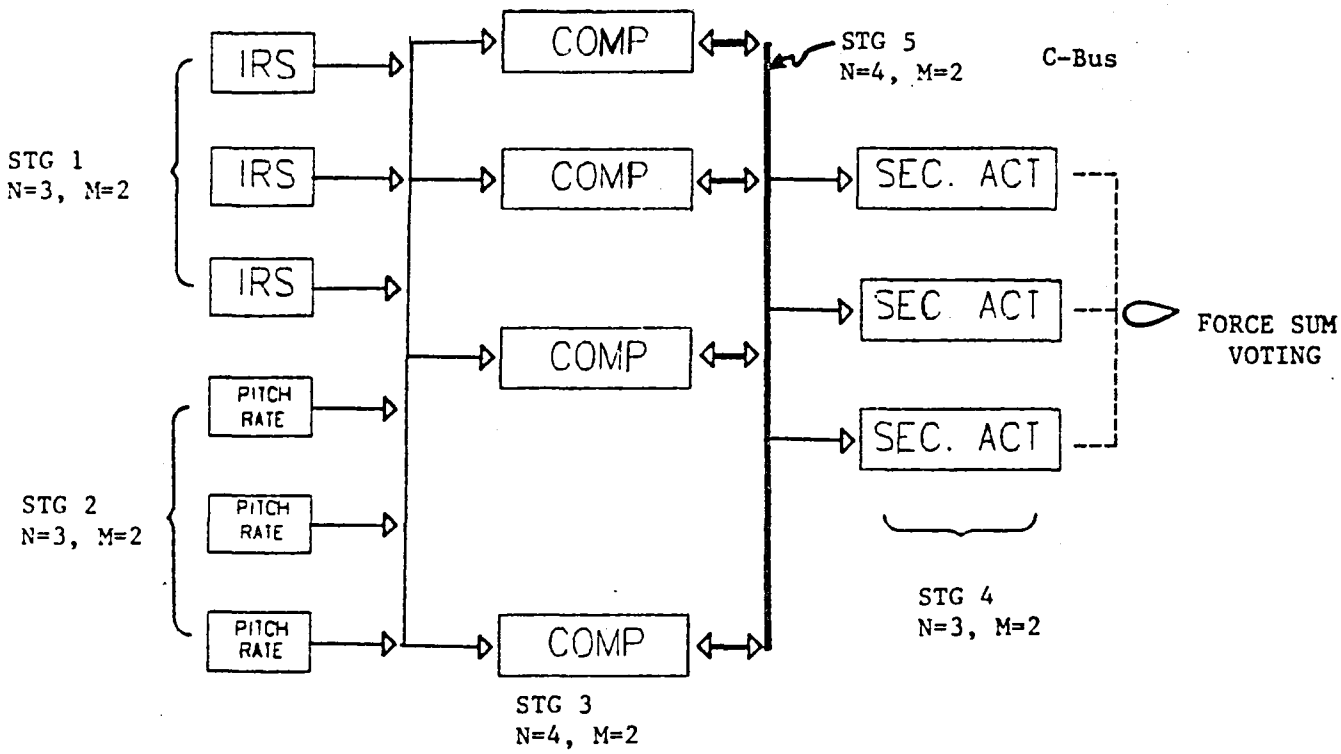
EXAMPLE PROBLEM 7

PROBLEM DESCRIPTION

Problem 7 includes critical-pair failures across stages as well as within a stage. To illustrate critical-pair failures across stages, a computer (C) bus stage which enables communication between computers is included. The C-bus stage is composed of four buses, but only two are required for reliable operation. Information going onto the buses are majority voted by the computers and information entering the receiving computers also majority vote the information.

One bus is a hot spare and not subject to critical-pair failure; however, the three in-use buses are. The spare bus is constantly flexed with the other three in a random fashion to expose latent failures. Since three voting processors communicate over three buses, a failure in a voting triad and a failure in a bus not connected to the failed computer would preclude a correct majority vote at the receiving end of the triplex bus. This cross coupling of failures in two different stages, i.e., computer and bus, is a critically coupled failure across stages.

FUNCTIONAL BLOCK DIAGRAM OF SYSTEM



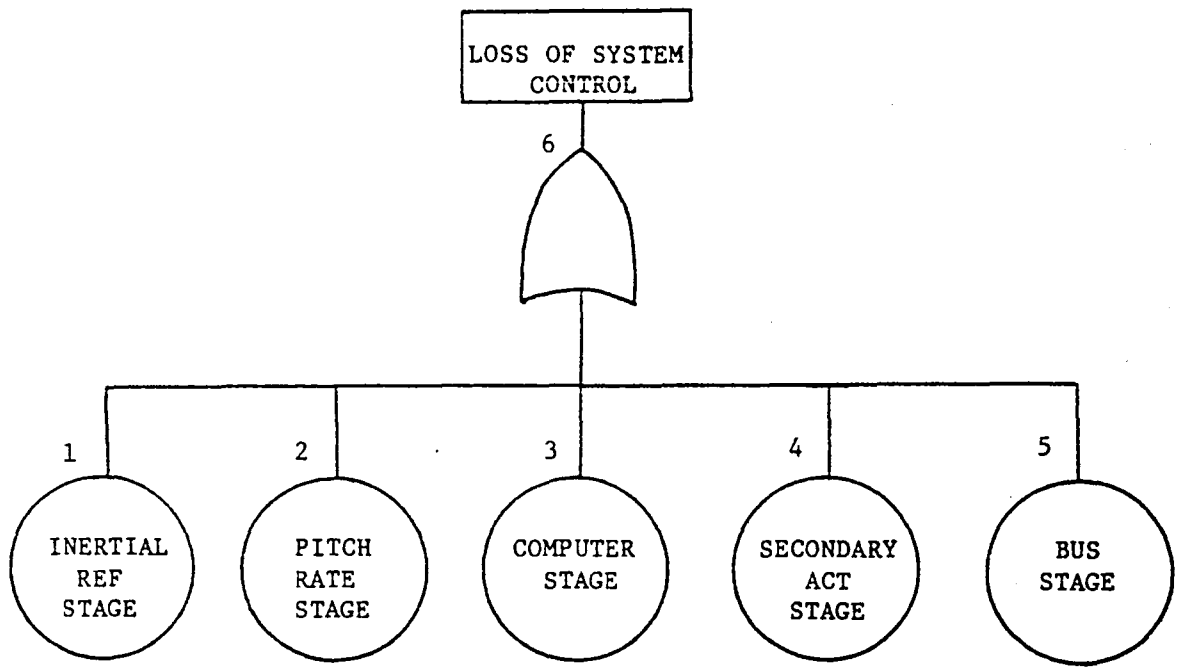
SYSTEM FAILURE CRITERIA

The system fails if any stage fails.
A single point failure in the computer stage causes system failure.
The system fails if a critically coupled failure occurs in the computer stage or bus stage.
The inertial reference stage fails if 2 out of 3 modules fail.
The computer stage fails if 3 out of 4 modules fail.
The secondary actuator fails if 2 out of 3 modules fail.

INPUT DATA FOR FAILURE OCCURRENCE MODEL

Parametric Data	Exponential
inertial reference sensor module	$\lambda_i = 1.5 \times 10^{-5}$
pitch rate sensor module	$\lambda_p = 1.9 \times 10^{-5}$
computer module	$\lambda_c = 4.8 \times 10^{-4}$
secondary actuator module	$\lambda_s = 3.7 \times 10^{-5}$
computer bus	$\lambda_b = 2.7 \times 10^{-6}$

SYSTEM TREE



N	3	3	4	3	4
M	2	2	2	2	2

INPUT DATA FOR SINGLE FAULT MODEL PARAMETRIC DATA

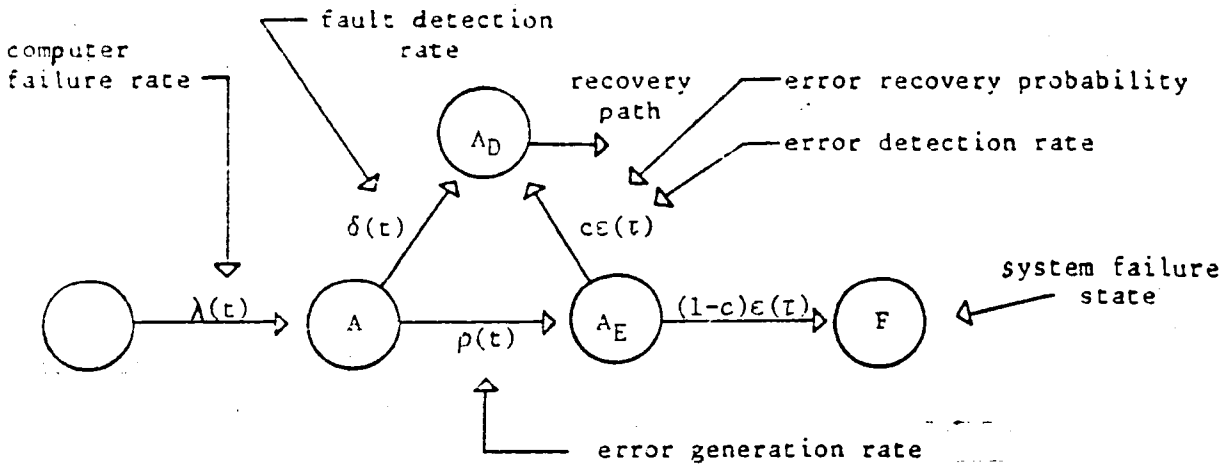
Self test rate $\delta(t)$ = 360 detections/hour for computer stage
$\delta(t) = 1.0 \times 10^4$ for computer bus stage
Random test = exponential detection times

Error detection rate $\varepsilon(t)$ = 3600 detections/hour for computer stage
$\varepsilon(t) = 0.0$ for computer bus stage
Random test = exponential

Error recovery probability C = 0.999 for computer stage
C = 1.0 for computer bus stage

Error generation rate $\rho(t)$ = 180 errors/hour for computer stage
$\rho(t) = 0.0$ for computer bus stage
Random pattern = exponential

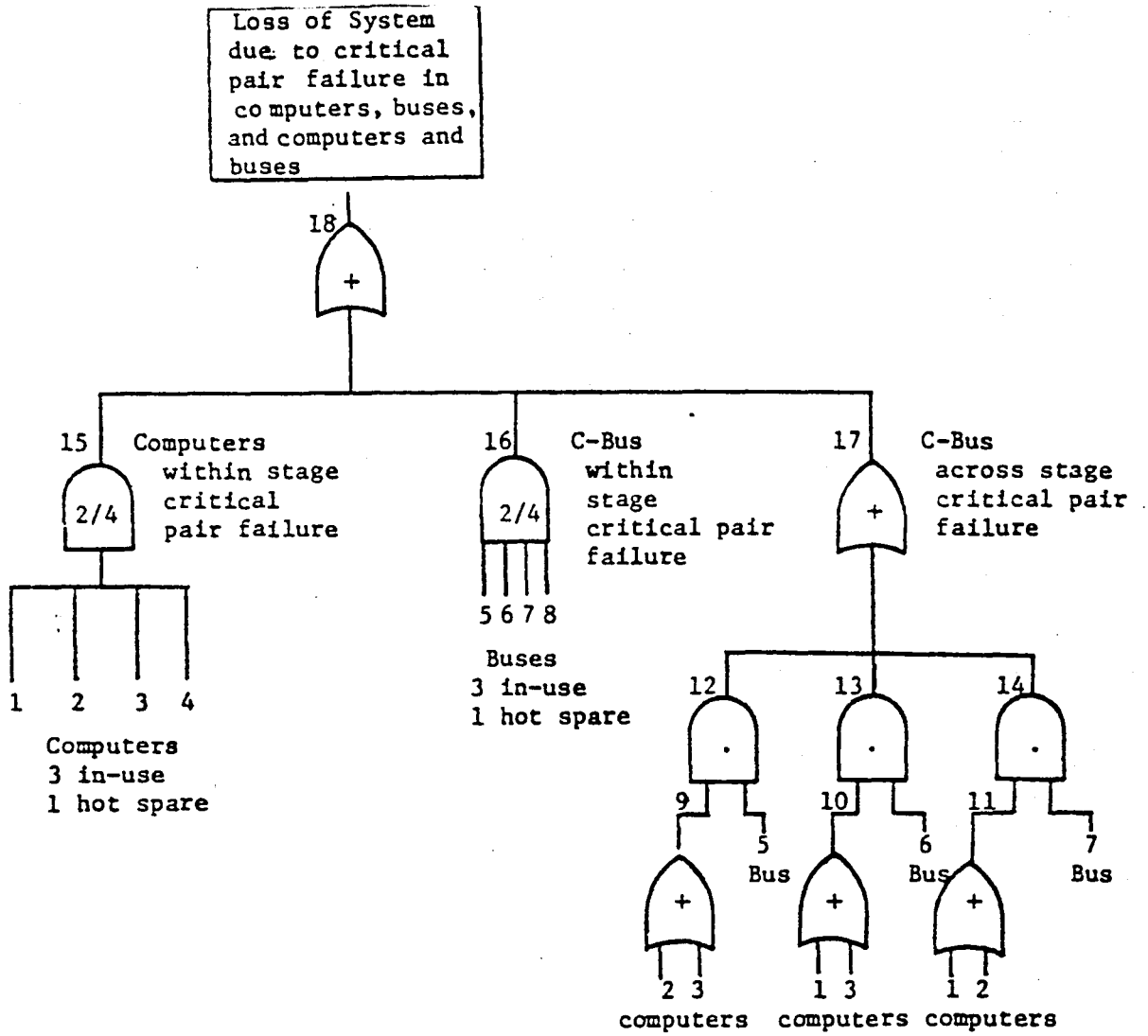
SINGLE FAULT MODEL



CRITICAL PAIR TREE

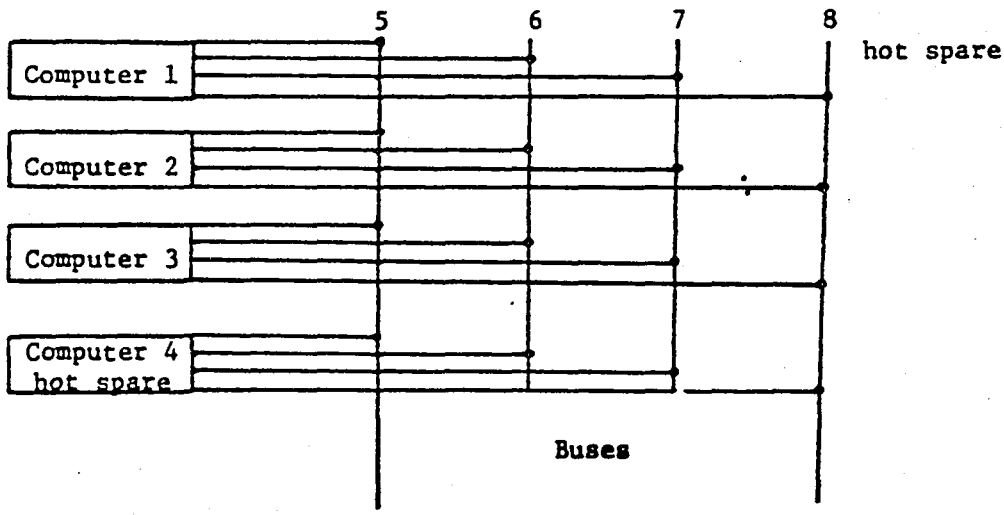
The 2/4 gate[1] with output 15 models the computer stage critically coupled failure. The 2/4 gate with output 16 models the bus stage critically coupled failure. The OR gate with output 17 and all lower connecting gates model critically coupled failures across the computer and bus stages. For example, if computer 2 or 3 fails and bus 5 fails, the bus will have bad data on 2 out of 3 in-use buses even though only one bus failed. The loss of the bus function causes system failure. Output 13 and 14 and lower level gates take into account the other combinations of a computer failure and a bus failure.

[1]Since NOP(1,3)=3 and NOP(1,5)=3, one module (4) in the computer stage and one module (8) in the bus stage are not critically coupled, i.e., they are spares, even though the critical-pair tree shows these modules in the 2/4 gates. The NOP specification overrides the gate specification. Equivalently, the gates could have been 2/3 gates without modules 4 and 8 included.



3 in-use computers always listen on 3 in-use buses.

Each computer talks on 1 unique bus, i.e. computer 1 talks on bus 5, computer 2 talks on bus 6, and computer 3 talks on bus 7

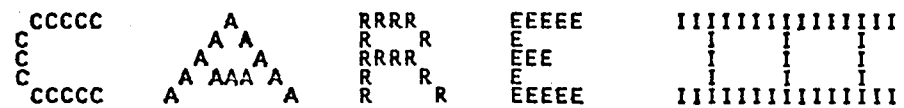


NAMELIST FORMAT INPUT

```
$FLTTYP
      NFTYPS=3,
      ALP=       Ø.Ø     ,       Ø.Ø     ,       Ø.Ø     ,
      BET=       Ø.Ø     ,       Ø.Ø     ,       Ø.Ø     ,
      DEL= 1.ØØØØØØE+Ø6,     36Ø.Ø     ,   1ØØØØ.Ø     ,
      RHO=       Ø.Ø     ,      18Ø.Ø     ,       Ø.Ø     ,
      EPS=       Ø.Ø     ,    36ØØ.Ø      ,       Ø.Ø     ,
      IDELF=       1     ,       1       ,       1       ,
      IRHOF=       1     ,       1       ,       1       ,
      IEPSF=       1     ,       1       ,       1       ,
      MARKOV=      1     ,
      PA=        1.Ø     ,       1.Ø     ,       1.Ø     ,
      PB=        Ø.Ø     ,       Ø.Ø     ,       Ø.Ø     ,
      C=         1.Ø     , 9.99ØØØØE-Ø1  ,       1.Ø     ,
      DBLDF= Ø.5ØØØØØE-Ø1,TRUNC= Ø.1ØØØØØE-Ø3,
      CVPRNT=F,CVPLOT=F,IAXSCV=   2$
$STAGES
      NSTGES=5,
      N =  3,  3,  4,  3,  4,
      M =  2,  2,  2,  2,  2,
      LC=  Ø,  Ø,  Ø,  Ø,  Ø,
      NOP(1,3)=3,
      NOP(1,5)=3,
      IRLPCD=4,
      RLPLOT=F          ,IAXSRL=2$
$FLTCAT
      NFCATS=1,1,1,1,1,
      JTYP(1,1)=  1,
      JTYP(1,2)=  1,
      JTYP(1,3)=  2,
      JTYP(1,4)=  1,
      JTYP(1,5)=  3,
      OMG(1,1)=       1.Ø     ,
      OMG(1,2)=       1.Ø     ,
      OMG(1,3)=       1.Ø     ,
      OMG(1,4)=       1.Ø     ,
      OMG(1,5)=       1.Ø     ,
      RLM(1,1)= 1.5ØØØØØE-Ø5,
      RLM(1,2)= 1.9ØØØØØE-Ø5,
      RLM(1,3)= 4.8ØØØØØE-Ø4,
      RLM(1,4)= 3.7ØØØØØE-Ø5,
      RLM(1,5)= 2.7ØØØØØE-Ø6,
$
$RNTIME
      FT=  1Ø.ØØØØ     ,ITBASE=1,
      SYSFLG=T,CPLFLG=T,
      NSTEPS=  5Ø,
      PSTRNC= Ø.1ØØØØØE-13,
      QPTRNC= Ø.1ØØØØØE-Ø1$
SYSTEM TREE EXAMPLE 7
1 5 6 6
6 O 1 2 3 4 5
CRITICAL PAIRS TREE EX 7
1 8 9 18
3 1 4
5 5 8
9 O 2 3
1Ø O 1 3
11 O 1 2
12 A 9 5
13 A 1Ø 6
14 A 11 7
15 2 1 2 3 4
16 2 5 6 7 8
17 O 12 13 14
18 O 15 16 17
```

LIST - DIRECTED FORMAT

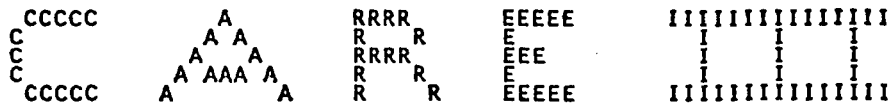
```
3,      Ø.Ø,      Ø.Ø,      Ø.Ø,
        Ø.Ø,      Ø.Ø,      Ø.Ø,
    1.ØE+6,    36Ø.Ø,   1ØØØØ.Ø,
        Ø.Ø,     18Ø.Ø,      Ø.Ø,
        Ø.Ø,    36ØØ.Ø,      Ø.Ø,
          1,         1,         1,
          1,         1,         1,
          1,         1,         1,
        1.Ø,       1.Ø,       1.Ø,
        Ø.Ø,       Ø.Ø,       Ø.Ø,
        1.Ø,      Ø.999,      1.Ø,
    Ø.Ø5, 1.ØE-4, .FALSE., .FALSE., 2, 1, /
5,    3, 3, 4, 3, 4,
      2, 2, 2, 2, 2,
      ,  ,  ,  ,  ,
      ,  ,  ,  ,  ,
      3, ,  ,  ,  ,
      ,  ,  ,  ,  ,
      3, ,  ,  ,  ,
      Ø, Ø, Ø, Ø, Ø,
      4, .FASLE., 2/
1, 1, 1, 1, 1,
          1,         1,         2,         1,         3,
        1.Ø,       1.Ø,       1.Ø,       1.Ø,       1.Ø,
    1.5E-5, 1.9E-5, 4.8E-4, 3.7E-5, 2.7E-6/
1Ø.Ø, 5Ø, 1, .TRUE., .TRUE., , 1.ØE-14, , Ø.Ø1, , , /
SYSTEM TREE EXAMPLE 7
1 5 6 6
6 O 1 2 3 4 5
CRITICAL PAIRS TREE EX 7
1 8 9 18
3 1 4
5 5 8
9 O 2 3
1Ø O 1 3
11 O 1 2
12 A 9 5
13 A 1Ø 6
14 A 11 7
15 2 1 2 3 4
16 2 5 6 7 8
17 O 12 13 14
18 O 15 16 17
```



STAGE NUMBER	PROBABILITY OF FAILURE AT 10.0 HOURS GIVEN PERFECT FAULT HANDLING
1	6.748311E-08
2	1.082658E-07
3	4.376183E-07
4	4.104468E-07
5	7.872730E-14

SUBRUN NUMBER	STARTING STAGE NUMBER	ENDING STAGE NUMBER	INVOLVES CRITICALLY COUPLED UNITS
1	1	2	F
2	3	5	T

SUCCESSFUL EXECUTION OF CAREIN.



C O V E R A G E F A U L T T Y P E S

D E N S I T Y F U N C T I O N S (GIVEN RATE R) :

1) R(T) = R*EXP(-R*T), FOR ALL T.

2) R(T) = R, FOR Ø.LE. T .LE.1/R,
= Ø, OTHERWISE.

S T E P S I Z E P A R A M E T E R S :

N S T E P S	F T	D B L D F	T R U N C
>= 5Ø	1.ØØØE+Ø1 HOURS	Ø.Ø5Ø	1.ØØØE-Ø4

TYPE:	RATES PER HOUR (DENSITY FUNCTION 1 OR 2) :					PROBABILITIES:		
ITYP	ALP	BET	DEL	RHO	EPS	PA	PB	C
1	Ø.ØØØE+ØØ (1)	Ø.ØØØE+ØØ (1)	1.ØØØE+Ø6 (1)	Ø.ØØØE+ØØ (1)	Ø.ØØØE+ØØ (1)	1.ØØØØØØ	Ø.ØØØØØØ	1.ØØØØØØ
2	Ø.ØØØE+ØØ (1)	Ø.ØØØE+ØØ (1)	3.6ØØE+Ø2 (1)	1.8ØØE+Ø2 (1)	3.6ØØE+Ø3 (1)	1.ØØØØØØ	Ø.ØØØØØØ	Ø.999ØØØ
3	Ø.ØØØE+ØØ (1)	Ø.ØØØE+ØØ (1)	1.ØØØE+Ø4 (1)	Ø.ØØØE+ØØ (1)	Ø.ØØØE+ØØ (1)	1.ØØØØØØ	Ø.ØØØØØØ	1.ØØØØØØ

S U B R U N 1

S T A G E S 1 - 2

C O N F I G U R A T I O N :			F A U L T	I N F O R M A T I O N :		
S T A G E	N	M	I C A T	R L M	O M G	J T V P
1	3	2	1	1.500E-05	1.00E+00	1
2	3	2	1	1.900E-05	1.00E+00	1

F A U L T V E C T O R S :

TIMES AT WHICH THE FOLLOWING FUNCTION VALUES CORRESPOND (IN HOURS):

0.000000000000E+00	3.052130 3415E-04	6.104260 6831E-04	9.156391 0246E-04	1.220852 1366E-03
1.831278 2049E-03	2.441704 2732E-03	3.052130 3415E-03	3.662556 4098E-03	4.883408 5464E-03
6.104260 6831E-03	7.325112 8197E-03	8.545964 9563E-03	1.098766 9230E-02	1.342937 3503E-02
1.587107 7776E-02	1.831278 2049E-02	2.319619 0596E-02	2.807959 9142E-02	3.296300 7689E-02
3.784641 6235E-02	4.761323 3328E-02	5.738005 0421E-02	6.714686 7514E-02	7.691368 4607E-02
9.644731 8792E-02	1.159809 5298E-01	1.355145 8716E-01	1.550482 2135E-01	1.941154 8972E-01
2.331827 5809E-01	2.722500 2646E-01	3.113172 9484E-01	3.894518 3158E-01	4.675863 6832E-01
5.457209 3487E-01	6.238554 7161E-01	7.801245 4510E-01	9.363936 1858E-01	1.092662 6921E+00
1.248931 7656E+00	1.561469 9125E+00	1.874000 8595E+00	2.186546 3257E+00	2.499084 4727E+00
3.124160 7666E+00	3.749237 0605E+00	4.374313 3545E+00	4.999389 6484E+00	6.249542 2363E+00
7.499694 8242E+00	8.749847 4121E+00	1.000000000000E+01		

** WARNING - CRITICAL FAULT PAIR FILE "BXYFL" DOES
NOT CONTAIN DATA FOR THIS SUBRUN. **

P(0, 0)

1.000000000000E+00	1.000000000000E+00	1.000000000000E+00	1.000000000000E+00	1.000000000000E+00
9.999998211 9E-01	9.999996423 7E-01	9.999996423 7E-01	9.999996423 7E-01	9.999994635 6E-01
9.999992847 4E-01	9.999992847 4E-01	9.999991059 3E-01	9.999987483 0E-01	9.999987483 0E-01
9.999983906 7E-01	9.999980033 5E-01	9.999976754 2E-01	9.999971389 8E-01	9.999966025 4E-01
9.999960660 9E-01	9.999951720 2E-01	9.999942779 5E-01	9.999932050 7E-01	9.999921321 9E-01
9.999901652 3E-01	9.999881982 8E-01	9.999862313 3E-01	9.999842643 7E-01	9.999801516 5E-01
9.999762177 5E-01	9.999721050 3E-01	9.999683499 3E-01	9.999603033 1E-01	9.999522566 8E-01
9.999443888 7E-01	9.999363422 4E-01	9.999204278 0E-01	9.999045133 6E-01	9.998885989 2E-01
9.998726844 8E-01	9.998406767 8E-01	9.998088479 0E-01	9.997770190 2E-01	9.997450113 3E-01
9.996813535 7E-01	9.996175766 0E-01	9.995539188 4E-01	9.994900082 6E-01	9.993626475 3E-01
9.992353916 2E-01	9.991078376 8E-01	9.989806413 7E-01		

Q(0, 0)

0.0 FOR ALL TIME STEPS.

P(0, 1)

0.000000000000E+00	1.739713 7952E-08	3.479429 7221E-08	5.219139 2541E-08	6.958862 2864E-08
1.043828 2771E-07	1.391770 0414E-07	1.739714 4347E-07	2.087657 2648E-07	2.783540 9355E-07

(CONTINUATION OF SUBRUN 1 DATA NOT LISTED)

SUBRUN 2

STAGES 3 - 5

CONFIGURATION:

STAGE	N	M
3	4	2
4	3	2
5	4	2

FAULT INFORMATION:

ICAT	RLM	OMG	JTYP
1	4.800E-04	1.00E+00	2
1	3.700E-05	1.00E+00	1
1	2.700E-06	1.00E+00	3

FAULT VECTORS:

TIMES AT WHICH THE FOLLOWING FUNCTION VALUES CORRESPOND (IN HOURS):

0.000000000000E+00	3.052130345E-04	6.104260683E-04	9.156391024E-04	1.220852136E-03
1.831278204E-03	2.441704273E-03	3.052130341E-03	3.662556409E-03	4.883408546E-03
6.104260683E-03	7.325112819E-03	8.545964956E-03	1.098766923E-02	1.342937350E-02
1.587107777E-02	1.831278204E-02	2.319619059E-02	2.807959914E-02	3.296300768E-02
3.784641623E-02	4.761323332E-02	5.738005042E-02	6.714686751E-02	7.691368460E-02
9.644731879E-02	1.159809529E-01	1.355145871E-01	1.550482213E-01	1.941154897E-01
2.331827588E-01	2.722500264E-01	3.113172948E-01	3.894518315E-01	4.675863832E-01
5.457209348E-01	6.238554716E-01	7.801245451E-01	9.363936185E-01	1.092662692E+00
1.248931766E+00	1.561469912E+00	1.874008059E+00	2.186546326E+00	2.499084472E+00
3.124160766E+00	3.749237060E+00	4.374313354E+00	4.999389648E+00	6.249542236E+00
7.499694824E+00	8.749847412E+00	1.000000000000E+01		

P(0, 0, 0)

1.000000000000E+00	9.999995231E-01	9.999988079E-01	9.999981523E-01	9.999974370E-01
9.999962449E-01	9.999948740E-01	9.999936819E-01	9.999927282E-01	9.999916523E-01
9.999876022E-01	9.999850392E-01	9.999826550E-01	9.999777674E-01	9.999725818E-01
9.999674558E-01	9.999627471E-01	9.999526739E-01	9.999428391E-01	9.999330044E-01
9.999226927E-01	9.999028444E-01	9.998826980E-01	9.998627901E-01	9.998431206E-01
9.998030662E-01	9.997632503E-01	9.997234344E-01	9.996838021E-01	9.996038079E-01
9.995237588E-01	9.994443655E-01	9.993644952E-01	9.992050528E-01	9.990450488E-01
9.988862872E-01	9.987272024E-01	9.984084963E-01	9.980900287E-01	9.977716207E-01
9.974530935E-01	9.968168139E-01	9.961809516E-01	9.955455064E-01	9.949104189E-01
9.936413765E-01	9.923739433E-01	9.911082983E-01	9.898443222E-01	9.873207211E-01
9.848037362E-01	9.822933674E-01	9.797890186E-01		

Q(0, 0, 0)

0.0 FOR ALL TIME STEPS.

P(0, 0, 1)

0.000000000000E+00	3.296298789E-09	6.592595803E-09	9.888877273E-09	1.318518006E-08
1.977772790E-08	2.637025531E-08	3.296280937E-08	3.955533507E-08	5.274028680E-08
6.592525637E-08	7.911009447E-08	9.229484022E-08	1.186641754E-07	1.450331837E-07

(CONTINUATION OF SUBRUN 2 DATA NOT LISTED)

SUMMARY INFORMATION:

Q SUM =

Ø.ØØØØØØØØØØØE+ØØ	4.6196648937E-12	2.7479836115E-11	7.3Ø257ıØ479E-11	1.3997712711E-1Ø
3.2846494968E-1Ø	5.7364Ø63499E-1Ø	8.5982443387E-1Ø	1.1754348606E-Ø9	1.8642647426E-Ø9
2.5981730101E-Ø9	3.3552958190E-Ø9	4.1246219773E-Ø9	5.6794315917E-Ø9	7.2415Ø33969E-Ø9
8.8Ø57126Ø35E-Ø9	1.Ø37Ø454717E-Ø8	1.35ØØ236484E-Ø8	1.663Ø632871E-Ø8	1.9761449366E-Ø8
2.2892326257E-Ø8	2.9154Ø24972E-Ø8	3.5415709476E-Ø8	4.1677363782E-Ø8	4.7938993220E-Ø8
6.Ø462170381E-Ø8	7.2985208988E-Ø8	8.55Ø8141Ø12E-Ø8	9.8Ø3Ø9593Ø5Ø-Ø8	1.23Ø7621944E-Ø7
1.4812103188E-Ø7	1.7316638958Ø-Ø7	1.982Ø926411E-Ø7	2.4829564945E-Ø7	2.983ØØØ3ØØ2E-Ø7
3.4846260633E-Ø7	3.9854324996E-Ø7	4.9869908Ø25E-Ø7	5.9884723669E-Ø7	6.98988ØØ348E-Ø7
7.9912109641E-Ø7	9.9936460174E-Ø7	1.1995781506E-Ø6	1.3997614587E-Ø6	1.5999145262E-Ø6
2.ØØØ1298253E-Ø6	2.4ØØ2242753E-Ø6	2.8ØØ197Ø8Ø4E-Ø6	3.2ØØØ4824Ø6E-Ø6	3.9993856262E-Ø6
4.7982348406E-Ø6	5.5965947467E-Ø6	6.3944644353E-Ø6		

P* SUM =

Ø.ØØØØØØØØØØØE+ØØ	5.4636653241E-16	2.1855184339E-15	4.9175217731E-15	8.7424845466E-15
1.9671457592E-14	3.4973Ø65432E-14	5.4648Ø14494E-14	7.8696767256E-14	1.3991Ø89285E-13
2.1864218877E-13	3.1487366967E-13	4.2861764485E-13	7.Ø866126552E-13	1.Ø588130850E-12
1.4791Ø82855E-12	1.9695911568E-12	3.1612583645E-12	4.6341272833E-12	6.3885112478E-12
8.4247Ø79798E-12	1.3343787880E-11	1.9393840700E-11	2.6577346227E-11	3.4896ØØØ727E-11
5.4953278Ø39E-11	7.9583124812E-11	1.Ø88Ø5985Ø6E-1Ø	1.4264173176E-1Ø	2.2423Ø67569E-1Ø
3.2450825360E-1Ø	4.4363240859E-1Ø	5.8176Ø41762E-1Ø	9.1566182414E-1Ø	1.3274764621E-Ø9
1.8184672657E-Ø9	2.3898958368E-Ø9	3.7791116902E-Ø9	5.5Ø52167269E-Ø9	7.5782997655E-Ø9
1.ØØØ8444740E-Ø8	1.5980152313E-Ø8	2.3500811963E-Ø8	3.2650731896E-Ø8	4.3510212322E-Ø8
7.Ø6787Ø9108E-Ø8	1.Ø564723141E-Ø7	1.4905222656E-Ø7	2.Ø1528791886-Ø7	3.3624664297E-Ø7
5.1484778396E-Ø7	7.4236572800E-Ø7	1.Ø2381Ø8953E-Ø6		

Q SUM + P* SUM =

Ø.ØØØØØØØØØØØE+ØØ	4.62Ø2113316E-12	2.7482Ø21867E-11	7.3Ø3Ø63Ø154E-11	1.3998587Ø12E-1Ø
3.2848462839E-1Ø	5.7367560702E-1Ø	8.5987905685E-1Ø	1.1755135754E-Ø9	1.8644Ø4637Ø7E-Ø9
2.5983917240E-Ø9	3.3556106782E-Ø9	4.1250505234E-Ø9	5.6801403581E-Ø9	7.2425621056E-Ø9
8.8Ø719142Ø6E-Ø9	1.Ø372424697E-Ø8	1.3503397511E-Ø8	1.6635267386E-Ø8	1.9767837145E-Ø8
2.29ØØ751517E-Ø8	2.9167368965E-Ø8	3.5435103740E-Ø8	4.1703941633E-Ø8	4.7973891526E-Ø8
6.Ø5171237Ø7E-Ø8	7.3Ø647897Ø4E-Ø8	8.5616946421E-Ø8	9.8173600804E-Ø8	1.233ØØ45251E-Ø7
1.4844553675E-Ø7	1.7360902405E-Ø7	1.9879102800E-Ø7	2.4921121167E-Ø7	2.9970749438E-Ø7
3.5Ø2810Ø8414E-Ø7	4.ØØ933146Ø4E-Ø7	5.Ø2478Ø2127E-Ø7	6.Ø4352469Ø6E-Ø7	7.Ø656631124E-Ø7
8.Ø912951717E-Ø7	1.Ø153447647E-Ø6	1.2230789252E-Ø6	1.4324122Ø48E-Ø6	1.6434247527E-Ø6
2.Ø7ØØØ84776E-Ø6	2.5Ø587140Ø3E-Ø6	2.9492493923E-Ø6	3.4Ø1577Ø325E-Ø6	4.3356321839E-Ø6
5.3130825108E-Ø6	6.3389607021E-Ø6	7.4182835306E-Ø6		

K STAGE FAILURES	PORTION OF UNRELIABILITY CAUSED BY FAULT HANDLING AFTER EXACTLY K STAGES HAVE FAILED BY 1.ØØØØE+Ø1 HOURS	PORTION OF UNRELIABILITY CAUSED BY EXHAUSTION OF MODULES OF EXACTLY K STAGES FAILING BY 1.ØØØØE+Ø1 HOURS
Ø	6.3944644353E-Ø6	Ø.ØØØØØØØØØØØE+ØØ
1	X	1.Ø2381886Ø9E-Ø6
2	X	3.3597476472E-13
3	X	3.7764349255E-2Ø
4	X	1.3123358863E-27
5	X	Ø.ØØØØØØØØØØØE+ØØ

TOTAL SYSTEM UNRELIABILITY AT 1Ø.Ø HOURS = 7.4182835306E-Ø6

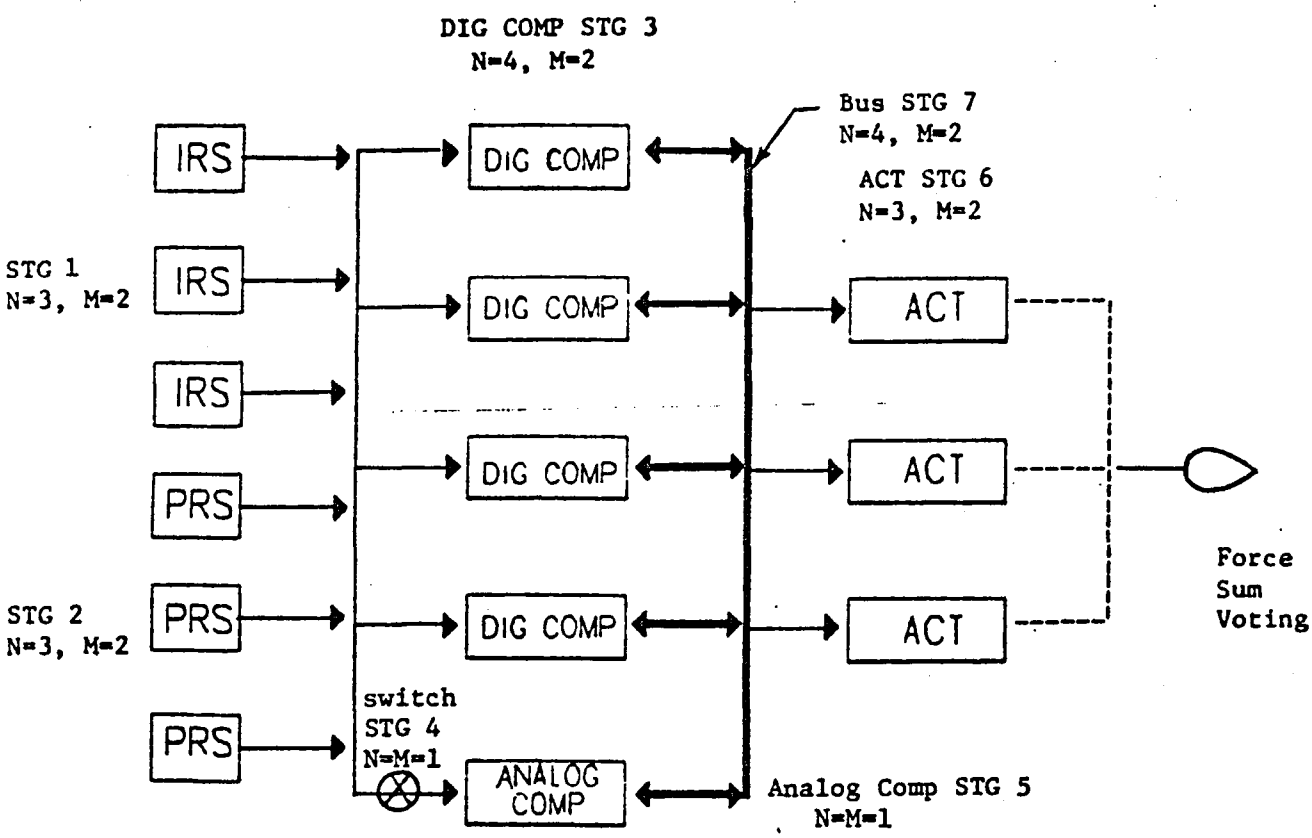
EXAMPLE PROBLEM 8

PROBLEM DESCRIPTION

Example 8 combines a critical-pair tree model based on example 7 with a system tree that incorporates stage redundancy. Since an analog computer is redundant to the digital computers, a critical-pair failure in the digital computers will not cause system failure. However, since the analog computer uses the computer bus, a critical-pair failure in the bus will cause system failure.

This example will also include a full single-fault model and adds a critical-pair intermittent model. The data for the latter model is given by the single-fault model and requires no further user input. The inclusion of an intermittent model in the single fault model ($\alpha > 0$, $\beta > 0$) in conjunction with a critical-pair tree allows the reliability assessment to take into account the effects of latent intermittents that can form critical pairs. More on this later. Single point failure for the computer stage is defined, but not for the bus stage, C = 1.0 for bus stage.

FUNCTIONAL BLOCK DIAGRAM OF SYSTEM



SYSTEM FAILURE CRITERIA:

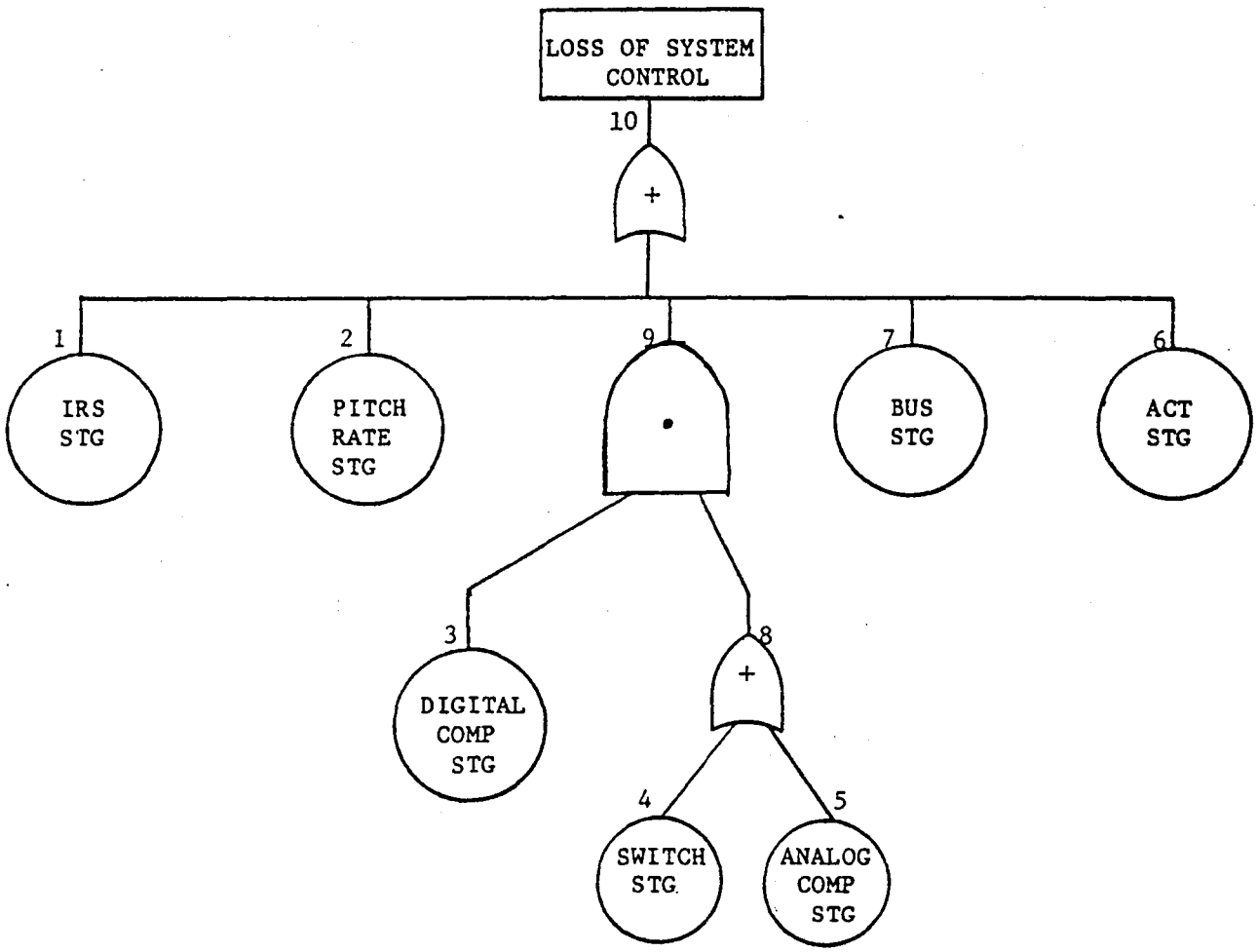
The system fails if the inertial reference, pitch rate, actuator stage, or bus stage fails, or the digital computer stage fails and the analog computer or transfer switch fails.

The system fails if a critically coupled failure occurs in the bus stage. The inertial reference stage fails if 2 out of 3 modules fail. The computer stage fails if 3 out of 4 modules fail. The secondary actuator fails if 2 out of 3 modules fail. The bus stage fails if 2 out of 3 modules fail.

INPUT DATA FOR FAILURE OCCURRENCE MODEL

Parametric Data	Exponential
inertial reference sensor module	$\lambda_i = 1.5 \times 10^{-5}$
pitch rate sensor module	$\lambda_p = 1.9 \times 10^{-5}$
digital computer module (permanent)	$\lambda_c = 4.8 \times 10^{-4}$
digital computer module (transient)	$\lambda_{ct} = 7.2 \times 10^{-3}$
digital computer module (intermittent)	$\lambda_{ci} = 3.3 \times 10^{-4}$
secondary actuator module	$\lambda_s = 3.7 \times 10^{-5}$
analog computer module	$\lambda_a = 2.3 \times 10^{-9}$
transfer switch	$\lambda_x = 1.7 \times 10^{-10}$
bus module (permanent)	$\lambda_b = 2.7 \times 10^{-6}$
bus module (transient)	$\lambda_{bt} = 6.2 \times 10^{-4}$
bus module (intermittent)	$\lambda_{bi} = 3.7 \times 10^{-5}$

SYSTEM TREE



INPUT DATA FOR SINGLE FAULT MODEL PARAMETRIC DATA

Self-test rate $\delta(t)$ = 360 detections/hour for computer, $\delta(t) = 1.0 \times 10^4$ for bus

Random test = exponential detection times for both

Error detection rate $\varepsilon(t)$ = 3600 detection/hour for computer, $\varepsilon(t) = 0.0$ for bus

Random test = exponential

Error recovery probability C = 1.0 for computer, C = 1.0 for bus

Error generation rate $\rho(t)$ = 180 errors/hour for computer, $\rho(t) = 0.0$ for bus

Random pattern = exponential

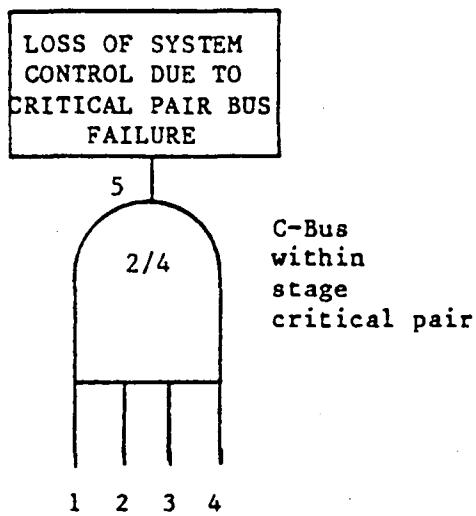
Transient duration rate $\alpha = 3.6 \times 10^4$ (exponential)
Intermittent duration rate $\alpha = 2.1 \times 10^3$ (exponential
Intermittent benign to active rate $\beta = 3.0 \times 10^3$ (exponential)
Retire module (fault) probability $P_A = 1.0$
Retire module (error) probability $P_B = 0.0$

} same for both

See figure 2.2 for the single fault-handling model.

CRITICAL-PAIR TREE

The critical-pair tree[1] does not include a gate for critical-pair failures in the computer stage because the analog computer is redundant to the digital computer stage. In other words, the loss of the digital computer stage, for whatever cause, will not cause system failure. The analog computer stage doesn't contribute a critical-pair gate linking the c-bus to the analog computer because the loss of the analog computer after a loss of the digital computer stage will cause system failure, irrespective of the bus failure. Only critical-pair failures in the bus stage will cause system failure.



3 in-use buses subject to critical-pair failures
1 hot spare not subject to critical-pair failures

The double intermittent fault model is invoked in the bus stage because the bus stage has intermittents defined and also a critical-pair failure tree (see fig. C-1 for double fault-handling). If an intermittent failure goes benign (see single-fault model) and another failure occurs that is critically coupled to the benign failure (critical-pair tree defined the linking), the double intermittent fault model is entered at state A_2B_1. Entry into state A_1B_2 is the inverse order case. Detection of the active fault takes the system to the detected state where the faulty active unit is purged. If the active fault at states A_2B_1 or A_1B_2 goes benign then state B_1B_2 is entered.

The last case takes the system to the failure state from A_2B_1 or A_1B_2 if the benign fault goes active or the active fault generates errors, or if the active fault is detected as nonpermanent (intermittent or transient). The failure condition occurs only if a critical-pair tree was defined and the single-fault model contains an intermittent fault model.

[1] The NOP(1,7)=3 specification overrides the 2/4 gate specification; effectively deleting module 4 as an input and treating the 2/4 gate as a 2/3 gate. In this case, a 2/3 gate with inputs 1, 2, and 3 would be equivalent to the above representation.

EXAMPLE 8 INPUT

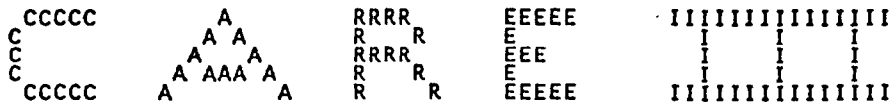
NAMELIST FORMAT

```
$FLTTYP
      NFTYPS=5,
      ALP=       Ø.Ø     ,       Ø.Ø     ,       Ø.Ø     ,   36ØØØ.Ø     ,    21ØØ.Ø     ,
      BET=       Ø.Ø     ,       Ø.Ø     ,       Ø.Ø     ,       Ø.Ø     ,    3ØØØ.Ø     ,
      DEL= 1.ØØØØØØE+Ø6,     36Ø.Ø     ,   1ØØØØ.Ø     ,     36Ø.Ø     ,     36Ø.Ø     ,
      RHO=       Ø.Ø     ,     18Ø.Ø     ,       Ø.Ø     ,     18Ø.Ø     ,     18Ø.Ø     ,
      EPS=       Ø.Ø     ,    36ØØ.Ø     ,       Ø.Ø     ,    36ØØ.Ø     ,    36ØØ.Ø     ,
      IDELF=       1     ,       1       ,       1       ,       1       ,       1       ,
      IRHOF=       1     ,       1       ,       1       ,       1       ,       1       ,
      IEPSF=       1     ,       1       ,       1       ,       1       ,       1       ,
      MARKOV=      1     ,
      PA=        1.Ø     ,       1.Ø     ,       1.Ø     ,       1.Ø     ,       1.Ø     ,
      PB=        Ø.Ø     ,       Ø.Ø     ,       Ø.Ø     ,       Ø.Ø     ,       Ø.Ø     ,
      C=         1.Ø     ,       1.Ø     ,       1.Ø     ,       1.Ø     ,       1.Ø     ,
      DBLDF= Ø.5ØØØØØE-Ø1,TRUNC= Ø.1ØØØØØE-Ø3,
      CVPRNT=F,CVPLOT=F,IAXSCV=   2$
$STAGES
      NSTGES=7,
      N =  3,  3,  4,  1,  1,  3,  4,
      M =  2,  2,  2,  1,  1,  2,  2,
      LC=  Ø,  Ø,  Ø,  Ø,  Ø,  Ø,  Ø,
      NOP(1,7)=3,
      IRLPCD=4,
      RLPLOT=F          ,IAXSRL=2$
$FLTCAT
      NFCATS=1,1,3,1,1,1,3,
      JTYP(1,1)=  1,
      JTYP(1,2)=  1,
      JTYP(1,3)=  4,  5,  2,
      JTYP(1,4)=  1,
      JTYP(1,5)=  1,
      JTYP(1,6)=  1,
      JTYP(1,7)=  4,  5,  3,
      OMG(1,1)=       1.Ø     ,
      OMG(1,2)=       1.Ø     ,
      OMG(1,3)=       1.Ø     ,       1.Ø     ,       1.Ø     ,
      OMG(1,4)=       1.Ø     ,
      OMG(1,5)=       1.Ø     ,
      OMG(1,6)=       1.Ø     ,
      OMG(1,7)=       1.Ø     ,       1.Ø     ,       1.Ø     ,
      RLM(1,1)= 1.5ØØØØØE-Ø5,
      RLM(1,2)= 1.9ØØØØØE-Ø5,
      RLM(1,3)= 7.2ØØØØØE-Ø3, 3.3ØØØØØE-Ø4, 4.8ØØØØØE-Ø4,
      RLM(1,4)= 1.7ØØØØØE-1Ø,
      RLM(1,5)= 2.3ØØØØØE-Ø9,
      RLM(1,6)= 3.7ØØØØØE-Ø5,
      RLM(1,7)= 6.2ØØØØØE-Ø4, 3.7ØØØØØE-Ø5, 2.7ØØØØØE-Ø6,
$
$RNTIME
      FT=  1Ø.ØØØØ     ,ITBASE=1,
      SYSFLG=T,CPLFLG=T,
      NSTEPS=  5Ø,
      PSTRNC= Ø.1ØØØØØE-13,
      QPTRNC= Ø.1ØØØØØE-Ø1$
SYSTEM TREE EX 8
1 7 8 1Ø
8 Ø 4 5
9 A 3 8
1Ø Ø 1 2 9 6 7
CRITICAL PAIRS TREE EX 8
1 4 5 5
7 1 4
5 2 1 2 3 4
```

EXAMPLE 3 INPUT

LIST-DIRECTED FORMAT

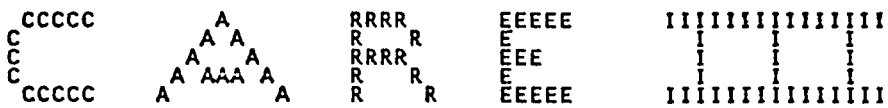
```
5,      Ø.Ø,       Ø.Ø,       Ø.Ø,   36ØØØ.Ø,    21ØØ.Ø,
        Ø.Ø,       Ø.Ø,       Ø.Ø,       Ø.Ø,    3ØØØ.Ø,
    1.ØE+6,     36Ø.Ø,   1ØØØØ.Ø,     36Ø.Ø,     36Ø.Ø,
        Ø.Ø,     18Ø.Ø,       Ø.Ø,     18Ø.Ø,     18Ø.Ø,
        Ø.Ø,    36ØØ.Ø,       Ø.Ø,    36ØØ.Ø,    36ØØ.Ø,
           1,         1,         1,         1,         1,
           1,         1,         1,         1,         1,
           1,         1,         1,         1,         1,
        1.Ø,       1.Ø,       1.Ø,       1.Ø,       1.Ø,
        Ø.Ø,       Ø.Ø,       Ø.Ø,       Ø.Ø,       Ø.Ø,
        1.Ø,       1.Ø,       1.Ø,       1.Ø,       1.Ø,
    Ø.ØØ5, 1.ØE-4, .FALSE., .FALSE., 2, 1, /
7,  3, 3, 4, 1, 1, 3, 4,
    2, 2, 2, 1, 1, 2, 2,
    3Ø*
    3, 1*,
    Ø, Ø, Ø, Ø, Ø, Ø, Ø,
    4, .FALSE., 2/
1, 1, 3, 1, 1, 1, 3,
           1,         1,         4,         5,         2,         1,         1,         1,
           4,         5,         3,
    11*1.Ø,
    1.5E-5, 1.9E-5, 7.2E-3, 3.3E-4, 4.8E-4, 1.7E-1Ø, 2.3E-9, 3.7E-5,
    6.2E-4, 3.7E-5, 2.7E-6/
1Ø.Ø, 5Ø, 1, .TRUE., .TRUE., , 1.ØE-14, , Ø.Ø1, , , /
 SYSTEM TREE EX 8
 1 7 8 1Ø
 8 Ø 4 5
 9 A 3 8
 1Ø Ø 1 2 9 6 7
 CRITICAL PAIRS TREE EX 8
 1 4 5 5
 7 1 4
 5 2 1 2 3 4
```



STAGE NUMBER	PROBABILITY OF FAILURE AT 10.0 HOURS GIVEN PERFECT FAULT HANDLING
1	6.748311E-08
2	1.082658E-07
3	2.087403E-06
4	1.699999E-09
5	2.299998E-08
6	4.104468E-07
7	2.500596E-10

SUBRUN NUMBER	STARTING STAGE NUMBER	ENDING STAGE NUMBER	INVOLVES CRITICALLY COUPLED UNITS
1	1	6	F
2	7	7	T

SUCCESSFUL EXECUTION OF CARE IN.



COVERAGE FAULT TYPES

DENSITY FUNCTIONS (GIVEN RATE R) :

1) R(T) = R*EXP(-R*T), FOR ALL T.

2) R(T) = R, FOR Ø.LE. T .LE.1/R,
= Ø, OTHERWISE.

STEP SIZE PARAMETERS :

NSTEPS	FT	DBLDF	TRUNC
>= 5Ø	1.ØØØE+Ø1 HOURS	Ø.Ø5Ø	1.ØØØE-Ø4

TYPE : ITYP	RATES PER HOUR (DENSITY FUNCTION 1 OR 2) : ALP	BET	DEL	RHO	EPS	PROBABILITIES : PA	PB	C
1	Ø.ØØØE+ØØ (1)	Ø.ØØØE+ØØ (1)	1.ØØØE+Ø6 (1)	Ø.ØØØE+ØØ (1)	Ø.ØØØE+ØØ (1)	1.ØØØØØØ	Ø.ØØØØØØ	1.ØØØØØØ
2	Ø.ØØØE+ØØ (1)	Ø.ØØØE+ØØ (1)	3.6ØØE+Ø2 (1)	1.8ØØE+Ø2 (1)	3.6ØØE+Ø3 (1)	1.ØØØØØØ	Ø.ØØØØØØ	1.ØØØØØØ
3	Ø.ØØØE+ØØ (1)	Ø.ØØØE+ØØ (1)	1.ØØØE+Ø4 (1)	Ø.ØØØE+ØØ (1)	Ø.ØØØE+ØØ (1)	1.ØØØØØØ	Ø.ØØØØØØ	1.ØØØØØØ
4	3.6ØØE+Ø4 (1)	Ø.ØØØE+ØØ (1)	3.6ØØE+Ø2 (1)	1.8ØØE+Ø2 (1)	3.6ØØE+Ø3 (1)	1.ØØØØØØ	Ø.ØØØØØØ	1.ØØØØØØ
5	2.1ØØE+Ø3 (1)	3.ØØØE+Ø3 (1)	3.6ØØE+Ø2 (1)	1.8ØØE+Ø2 (1)	3.6ØØE+Ø3 (1)	1.ØØØØØØ	Ø.ØØØØØØ	1.ØØØØØØ

SUBRUN 1

STAGES 1 - 6

CONFIGURATION:

STAGE	N	M
1	3	2
2	3	2
3	4	2
4	1	1
5	1	1
6	3	2

FAULT INFORMATION:

ICAT	RLM	OMG	JTYP
1	1.500E-05	1.00E+00	1
1	1.900E-05	1.00E+00	1
1	7.200E-03	1.00E+00	4
2	3.300E-04	1.00E+00	5
3	4.800E-04	1.00E+00	2
1	1.700E-10	1.00E+00	1
1	2.300E-09	1.00E+00	1
1	3.700E-05	1.00E+00	1

FAULT VECTORS:

TIMES AT WHICH THE FOLLOWING FUNCTION VALUES CORRESPOND (IN HOURS):

0.000000000000E+00	9.770039569 99E-04	1.954007914 00E-03	2.931118710 0E-03	3.900815828 00E-03
4.885198008 28E-03	6.839277222 8E-03	8.793356828 4E-03	1.007474364 34E-02	1.270151604 0E-02
1.465559564 5E-02	1.856375485 7E-02	2.247191407 68E-02	2.638007327 9E-02	3.028823249 0E-02
3.419639170 2E-02	4.201271012 4E-02	4.982902854 7E-02	5.764534696 9E-02	6.546166539 2E-02
7.327798000 9E-02	8.891006169 34E-02	1.004543253 78E-01	1.201758906 2E-01	1.358008527 47E-01
1.514411568 6E-01	1.827064300 55E-01	2.139717042 4E-01	2.452369779 3E-01	2.765022516 3E-01
3.077675104 1E-01	3.702980057 79E-01	4.328286005 18E-01	4.953591525 6E-01	5.578896999 4E-01
6.204202175 1E-01	7.454813122 7E-01	8.705424070 4E-01	9.956035018 0E-01	1.120664596 6E+00
1.245725631 7E+00	1.495847821 2E+00	1.745970010 8E+00	1.996092200 3E+00	2.246214389 8E+00
2.496336460 1E+00	2.996580083 92E+00	3.496825218 2E+00	3.997069597 2E+00	4.497313976 3E+00
4.997558116 9E+00	5.998046875 0E+00	6.998535633 1E+00	7.999024391 2E+00	8.999512672 4E+00
1.000000009 54E+01				

WARNING - CRITICAL FAULT PAIR FILE "BXYFL" DOES
CONTAIN DATA FOR THIS SUBRUN. **

P(0, 0, 0, 0, 0, 0)

1.000000000000E+00	9.999964833 3E-01	9.999927282 3E-01	9.999890327 5E-01	9.999856948 9E-01
9.999819994 0E-01	9.999744892 1E-01	9.999672174 5E-01	9.999595284 5E-01	9.999527335 2E-01
9.999452829 4E-01	9.999305605 9E-01	9.999160170 6E-01	9.999017119 4E-01	9.998865723 6E-01
9.998722672 5E-01	9.998430013 7E-01	9.998135566 7E-01	9.997845292 1E-01	9.997555017 5E-01
9.997261762 6E-01	9.996679425 2E-01	9.996092912 9E-01	9.995508194 0E-01	9.994922876 4E-01
9.994338154 8E-01	9.993171095 8E-01	9.992000460 6E-01	9.990833401 7E-01	9.989662766 5E-01
9.988498091 7E-01	9.986160993 6E-01	9.983828067 8E-01	9.981493949 9E-01	9.979162216 2E-01

(CONTINUATION OF SUBRUN 1 DATA NOT LISTED)

SUBRUN 2

STAGES 7 - 7

CONFIGURATION:

STAGE	N	M
7	4	2

FAULT INFORMATION:

ICAT	RLM	OMG	JTYP
1	6.200E-04	1.00E+00	4
2	3.700E-05	1.00E+00	5
3	2.700E-06	1.00E+00	3

FAULT VECTORS:

TIMES AT WHICH THE FOLLOWING FUNCTION VALUES CORRESPOND (IN HOURS):

0.000000000000E+00	9.770395699 9E-04	1.954079140 0E-03	2.931118710 0E-03	3.908158280 0E-03
4.885198008 28E-03	6.839277222 8E-03	8.793356828 4E-03	1.074743643 4E-02	1.270151640 E-02
1.465559564 5E-02	1.856375485 7E-02	2.247191406 8E-02	2.638007327 9E-02	3.028823249 0E-02
3.419639170 2E-02	4.201271012 4E-02	4.982902855 47E-02	5.764534696 9E-02	6.546166539 2E-02
7.327798000 89E-02	8.891066169 34E-02	1.045432537 8E-01	1.201758906 2E-01	1.358085274 7E-01
1.514411568 6E-01	1.827064300 55E-01	2.139717042 4E-01	2.452369779 3E-01	2.765022516 3E-01
3.077675104 1E-01	3.702980577 9E-01	4.328286050 8E-01	4.953591525 6E-01	5.578896999 4E-01
6.204202175 1E-01	7.454813122 7E-01	8.705424070 4E-01	9.956035018 0E-01	1.120664596 6E+00
1.245725631 7E+00	1.495847821 2E+00	1.745970010 8E+00	1.996092200 3E+00	2.246214389 8E+00
2.496336460 1E+00	2.996580839 2E+00	3.496825218 2E+00	3.997069597 2E+00	4.497313976 3E+00
4.997558116 9E+00	5.998046875 0E+00	6.998535633 1E+00	7.999024391 2E+00	8.999512672 4E+00
1.000000009 54E+01				

P(0)

1.000000000000E+00	9.999997615 8E-01	9.999997615 8E-01	9.999995231 6E-01	9.999995231 6E-01
9.999992847 4E-01	9.999988079 1E-01	9.999985694 9E-01	9.999980926 5E-01	9.999976158 1E-01
9.999973774 0E-01	9.999964237 2E-01	9.999959468 8E-01	9.999952316 3E-01	9.999945163 7E-01
9.999935627 0E-01	9.999923706 1E-01	9.999909400 9E-01	9.999892711 6E-01	9.999878406 5E-01
9.999866485 6E-01	9.999837875 4E-01	9.999806881 0E-01	9.999780654 9E-01	9.999752044 7E-01
9.999721050 3E-01	9.999666214 0E-01	9.999606606 3E-01	9.999551773 1E-01	9.999489784 2E-01
9.999434948 0E-01	9.999320507 0E-01	9.999203681 9E-01	9.999094000 4E-01	9.998977184 3E-01
9.998862743 4E-01	9.998631477 4E-01	9.998404979 7E-01	9.998173713 7E-01	9.997942447 7E-01
9.997715950 0E-01	9.997253418 0E-01	9.996795654 3E-01	9.996338486 7E-01	9.995880723 0E-01
9.995420575 1E-01	9.994502663 6E-01	9.993585348 1E-01	9.992666862 7E-01	9.991751323 2E-01
9.990836381 9E-01	9.988998770 7E-01	9.987167716 0E-01	9.985333085 1E-01	9.983499646 2E-01
9.981669187 5E-01				

Q(0)

0.000000000000E+00	1.342237885 0E-14	3.467526826 7E-14	5.502876596 8E-14	7.563945265 5E-14
9.625016305 9E-14	1.374715635 4E-13	1.786929572 4E-13	2.199143373 9E-13	2.611356768 8E-13
3.023570163 8E-13	3.847996682 6E-13	4.672422388 2E-13	5.496847551 8E-13	6.321272173 2E-13
7.145696252 6E-13	8.794542242 9E-13	1.044338660 7E-12	1.209222826 0E-12	1.374106666 1E-12
1.538990289 4E-12	1.868757102 2E-12	2.198522830 8E-12	2.528287475 2E-12	2.858051252 3E-12
3.187813945 1E-12	3.847336945 6E-12	4.506856259 7E-12	5.166371670 8E-12	5.825883178 7E-12

(CONTINUATION OF SUBRUN 2 DATA NOT LISTED)

SUMMARY INFORMATION:

Q SUM =

0.000000000000E+00	3.5457161702E-14	1.0982400834E-13	2.0484755927E-13	3.1653228216E-13
4.4071446796E-13	7.1606577005E-13	1.0147201005E-12	1.3267592320E-12	1.6466360610E-12
1.9710802001E-12	2.6269281889E-12	3.2867870751E-12	3.9479934079E-12	4.6097080146E-12
5.2716212472E-12	6.5955418210E-12	7.9194793084E-12	9.2434957258E-12	1.0567568522E-11
1.1891650858E-11	1.4539813797E-11	1.7187980206E-11	1.9836146614E-11	2.2484314757E-11
2.5132484635E-11	3.0428826125E-11	3.5725172820E-11	4.1021522984E-11	4.6317880087E-11
5.1614237190E-11	6.2206968743E-11	7.2799717643E-11	8.3392480421E-11	9.3985270955E-11
1.0457807537E-10	1.2576373276E-10	1.4694946648E-10	1.6813528347E-10	1.8932115597E-10
2.1050708399E-10	2.5287918981E-10	2.9525154543E-10	3.3762420316E-10	3.7999720037E-10
4.2237041642E-10	5.0711768118E-10	5.9186616719E-10	6.7661576342E-10	7.6136646987E-10
8.4611828655E-10	1.0156253616E-09	1.1851369885E-09	1.3546531674E-09	1.5241736762E-09
1.6936987368E-09				

P* SUM =

0.000000000000E+00	5.5987632675E-15	2.2395075093E-14	5.0388844844E-14	8.9580252938E-14
1.3996909101E-13	2.7433963304E-13	4.5350003050E-13	6.7745109652E-13	9.4619223481E-13
1.2597232285E-12	2.0211575469E-12	2.9617484680E-12	4.0815042318E-12	5.3804183331E-12
6.8584938076E-12	1.0352135815E-11	1.4562399897E-11	1.9489318145E-11	2.5132885356E-11
3.1493110203E-11	4.6363433925E-11	6.4100433295E-11	8.4703986880E-11	1.0817396284E-10
1.3451070813E-10	1.9578377808E-10	2.6852348123E-10	3.5272931798E-10	4.4840176017E-10
5.5554005840E-10	8.0421630466E-10	1.0987589727E-09	1.4391688952E-09	1.8254416867E-09
2.2575836756E-09	3.2594633659E-09	4.4448107417E-09	5.8136238046E-09	7.3659065514E-09
9.1016492121E-09	1.3123560016E-08	1.7879358438E-08	2.3369068458E-08	2.9592639450E-08
3.6550172666E-08	5.2666955241E-08	7.1719483685E-08	9.3707811288E-08	1.1863201621E-07
1.4649199898E-07	2.1102023595E-07	2.8729249379E-07	3.7531023622E-07	4.7507262479E-07
5.8658127955E-07				

Q SUM + P* SUM =

0.000000000000E+00	4.1055924123E-14	1.3221907835E-13	2.5523639734E-13	4.0611253509E-13
5.8068355896E-13	9.9040545730E-13	1.4682201852E-12	2.0042103827E-12	2.5928282958E-12
3.2308034286E-12	4.6480857357E-12	6.2485355432E-12	8.0294980734E-12	9.9901267814E-12
1.2130115488E-11	1.6947677636E-11	2.2481879206E-11	2.8732814739E-11	3.5700453010E-11
4.3384761061E-11	6.0903247723E-11	8.1288413500E-11	1.0454013349E-10	1.3065827587E-10
1.5964318756E-10	2.2621260420E-10	3.0424865405E-10	3.9375083749E-10	4.9471965413E-10
6.0715427130E-10	8.6642326647E-10	1.1715587389E-09	1.5225614103E-09	1.9194270617E-09
2.3621617995E-09	3.3852272097E-09	4.5917603053E-09	5.9817590881E-09	7.5552275547E-09
9.3121563793E-09	1.3376439512E-08	1.8174610261E-08	2.3706693497E-08	2.9726359296E-08
3.6972544137E-08	5.3174073145E-08	7.2311351573E-08	9.4384425608E-08	1.1939337696E-07
1.4733811327E-07	2.1203585732E-07	2.8847762223E-07	3.7666490016E-07	4.7659679581E-07
5.8827498606E-07				

K STAGE FAILURES	PORTION OF UNRELIABILITY CAUSED BY FAULT HANDLING AFTER EXACTLY K STAGES HAVE FAILED BY 1.0000E+01 HOURS	PORTION OF UNRELIABILITY CAUSED BY EXHAUSTION OF MODULES OF EXACTLY K STAGES FAILING BY 1.0000E+01 HOURS
0	1.6936987368E-09	0.000000000000E+00
1	X	5.8657963109E-07
2	X	1.7369058665E-12
3	X	2.5815965706E-19
4	X	1.3570143988E-26
5	X	2.1270726398E-34
6	X	0.000000000000E+00
7	X	0.000000000000E+00

TOTAL SYSTEM UNRELIABILITY AT 10.0 HOURS = 5.8827498606E-07

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA TM-86404		

4. Title and Subtitle	5. Report Date
CARE III MODEL OVERVIEW AND USER'S GUIDE (first revision)	April 1985
	6. Performing Organization Code 505-34-13-30
7. Author(s) S. J. Bavuso, P. L. Petersen	8. Performing Organization Report No.
	10. Work Unit No.
9. Performing Organization Name and Address: NASA Langley Research Center, Hampton, Virginia 23665	11. Contract or Grant No.
	13. Type of Report and Period Covered: Technical Memorandum
12. Sponsoring Agency Name and Address: National Aeronautics and Space Administration, Washington, DC 20546	14. Sponsoring Agency Code

15. Supplementary Notes: P. L. Petersen is an employee of Kentron International, Inc. This document supercedes NASA CR-165864, CARE III Phase II Report - User's Manual, Nov. 1982 and NASA TM-85810, June 1984.

16. Abstract

This document was written to introduce the CARE III (Computer-Aided Reliability Estimation) capability to reliability and design engineers who are interested in predicting the reliability of highly reliable fault-tolerant systems. It was also structured to serve as a quick-look reference manual for more experienced users.

The guide covers CARE III modeling and reliability predictions for execution in the CDC CYBER 170 series computers, DEC VAX-11/700 series computer, and most machines that compile ANSI Standard Fortran 77.

17. Key Words (Suggested by Author(s)): CARE III, Reliability, Fault-tolerance, Coverage, Fault handling

18. Distribution Statement: Unclassified - Unlimited

Subject category 61

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	149	A07

N-305

For sale by the National Technical Information Service, Springfield, Virginia 22161

End of Document